Secure Access Delegation of Encrypted Medical Information

Arnab Deb Gupta, Yuriy Polyakov, and Kurt Rohloff
College of Computing Sciences
New Jersey Institute of Technology
Newark, NJ 07102
Email: {ad479, polyakov, rohloff}@njit.edu

***Abstract*—The design of modern medical data information systems is driven by the need to collect and present data to authorized users. For collected medical data to be effective and improve patient treatment it must be transported from a device, aggregated, and analyzed to produce results that can be shared with care providers. Medical data may be analyzed and used years after collection at different locations because data sources and care providers often operate on different time scales and are geographically distributed. The need for distributed and long-term medical data storage thus requires an effective security model to delegate data access. Current data access delegation models do not provide end-to-end protection. An effective delegation model must keep data encrypted at all times and avoid the need to share decryption keys to avoid security vulnerabilities. We present a secure information architecture and prototype to implement such a model with end-to-end data encryption while restricting data access to designated recipients. Our architecture integrates recent Proxy Re-Encryption (PRE) advances into a client-server based security model that can be applied to open Internet communications. We discuss design tradeoffs and show experimental results. Our architecture lowers health care data management costs by enabling the secure outsourcing of data hosting to low-cost cloud computing environments. The architecture will also reduce the vulnerability of health care data systems to security challenges such as attacks compromising confidentiality and malicious insiders.**

I. Introduction

The secure collection, storage, and use of medical data is a relevant requirement and challenge to the health care system. Health care involves a diverse set of data collection systems such as medical records, health surveys, administrative records, and even wearable devices that reflect a patient's current location and level of health. This data is collected and used by various entities such as hospitals, health care centers, physicians, and health plans and the data is often collected in different locations from where it's used. These complexities and variations in data collection and processing are amplified by occurring over different time scales. For example, data may be collected every few seconds in the case of life-threatening emergency alerts or used years after collection in the case of pediatric data. Regardless of location or time, medical data is highly regulated and medical data security is a pressing concern [1].

In order to fully realize the benefits of medical data, a medical data security model must simultaneously be accessed efficiently, encourage rapid learning, and protect patient privacy. However, the geographically and temporally distributed lifecycles and security concerns attached to medical data make it challenging to create low-cost information architectures that allow secure data sharing. It is difficult to balance security, effectiveness, and cost concerns because they are often co-dependent and tradeoffs between them may not be desired. Current security models protect medical data with encryption technology when data is in motion (i.e. when transmitted between storage points) or at rest (i.e. when stored). However, the sharing of medical data requires either decrypting the data or the sender and recipient sharing decryption keys. These limitations compel medical data processing to occur in trusted computing environments. These environments can be expensive to set up and maintain because they require specialized management and housing in dedicated facilities. An additional issue with current security models is that data is only encrypted when it is sent to a pre-approved recipient. This point-to-point transmission time encryption requires prior coordination of decryption keys. That coordination requirement creates a security vulnerability data thieves can exploit to steal the keys. Security and effectiveness trade-offs have prevented the widespread use of low-cost cloud computing environments [2] and significant engineering efforts have been needed to securely integrate health care systems with larger information ecosystems [3]. The preceding concerns and deficiencies raise the cost of patient treatment and reduce the flexibility and effectiveness of collected medical data. Additionally, it has been difficult, if not impossible, to use scalable information architectures in low-cost cloud computing environments because of the difficulty of sharing encrypted data.

We have created a secure information architecture for medical data that eliminates current security model deficiencies by using end-to-end lattice encryption to 1) transmit encrypted data to a cloud computing environment and 2) share data with recipients without having to decrypt the data or share decryption keys. Lattice encryption is a relatively new family of encryption technologies [4] based on the hardness of variants of the "Shortest Vector Problem" [5]. Lattice encryption schemes are considered post-quantum and are secure against attacks from adversaries with both practical quantum and classical computing devices [6]. This means that lattice encryption schemes such as the one our architecture is based on [7] are more future-proof than legacy public key encryption

schemes such as RSA which can be defeated by quantum computing devices.

The contribution of this paper is an end-to-end secure information architecture for the collection, processing, and distribution of encrypted medical data using a lattice-based variant of Proxy Re-Encryption (PRE). We describe our architecture's prototype implementation which is a web-based solution allowing patients to publish encrypted data to a PRE-enabled server and doctors to securely access, share, and use data through that server. We discuss the engineering and design considerations accounted for in developing the prototype. We also evaluate the prototype under the CIA triad (Confidentiality, Integrity, and Availability) and suggest future steps to better support those evaluation factors.

This paper is organized as follows. In Section II we examine the security model and identify functional, security, and cost features that should be considered when evaluating our architecture. We discuss how we can use PRE to securely delegate access to encrypted data in Section III. We present our architecture's prototype implementation in Section IV. Section V discusses our architecture's initial experimental results. Section VI discusses related encrypted computing activities and applications. The paper concludes with a discussion of our insights and cost reduction benefits enabled by our architecture in Section VII.

II. Security Model Use Case

Figure 1 provides a use case for applying a PRE security model to a medical scenario. The use case's starting point is the ciphertext produced by encrypting a patient's data and storing it on their doctor's cloud-based data store (i.e., 1st Doctor Store). A patient may change doctors thereby requiring the new doctor to have access to the patient's data. Instead of the first doctor downloading and decrypting the patient's data and then encrypting the data again for the second doctor's data store, a PRE model allows the first doctor to delegate access to the patient's encrypted data in re-encrypted form on the originating data store. In order for this to occur, the second doctor generates their own public/private key pair and then the first doctor uses their private key and the second doctor's public key to generate a "re-encryption" key. This re-encryption key is sent to the first doctor's cloud provider (i.e., 1st Doctor Store). The cloud provider uses the re-encryption key for a PRE operation to convert the patient's encrypted data into a new ciphertext that can only be decrypted with the second doctor's private key. No cloud provider can use a re-encryption key to decrypt any data it stores. The PRE model described in this Section thus allows the first doctor to securely delegate decryption access to the second doctor for the patient's data without decrypting (or taxing bandwidth by downloading) that data. This model can be extended to a multi-hop PRE scheme for delegating access to any number of doctors by generating a re-encryption key for each recipient and then re-encrypting the patient's data so it can only be decrypted by each recipient's private key. We envision this model being implemented as a middleware with adaptability

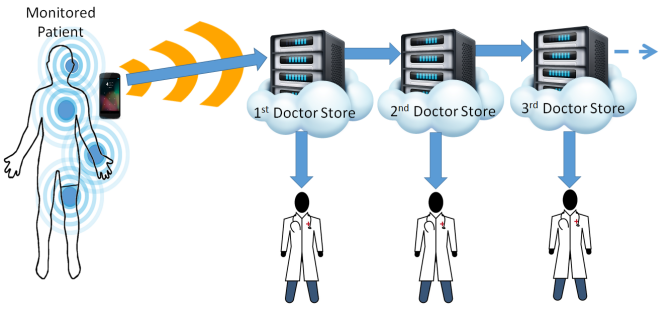

Fig. 1. Information Architecture Use Case

to underlying communication protocols and upper-layer data manipulation applications.

Standard information security frameworks evaluate systems under the Confidentiality, Integrity, and Availability (CIA) triad. We particularly seek to address Confidentiality by enabling a cloud-based data store to 1) delegate data access without exposing unencrypted data or decryption keys and 2) grant access to the encrypted data without being able to grant access to themselves.

III. Secure Access Delegation

Lattice-based Proxy Re-Encryption (PRE) provides an effective approach to delegate decryption ability. There are existing PRE designs based on lattice encryption [9]. As explained in [9], that recent lattice-based PRE scheme uses nearly identical algorithms for key generation, encryption, and decryption as the Homomorphic Encryption (HE) scheme in [7] thereby allowing use of similar proofs of security.

The primary implementation difference between HE and PRE is one of parameterization. Analysis from [9] indicates that the PRE scheme we present can be parameterized to use ring dimensions of $n = 1024$ and ciphertext moduli of less than 32 bits to encrypt 1024 plaintext bits. Experimentally, these parameterizations result in 1) the isolated re-encryption process running in under 40 ms per kb of data and 2) the encryption and decryption processes running in 6.1 and 7.9 ms per kb of data. These results demonstrate the feasibility of using the PRE model for practical medical data applications and even for time-critical medical performance requirements.

IV. Secure Information Architecture

Our architecture's prototype implementation is a Java-based web solution allowing 1) a patient to publish encrypted data to a PRE-enabled server and 2) a doctor to securely access, use, and delegate data through that server. We implemented the prototype's lattice-based PRE cryptosystem in C++ and its web solution using Java Server Pages and Java Servlets running in Apache Tomcat 8.0. Additionally, we created a Java-based Policy Server to distribute public keys to users. The prototype uses a MySQL Server 5.6 instance to store encrypted and re-encrypted patient data on the PRE Server.

We selected C++ for the prototype's PRE cryptosystem because of its cross-platform support, performance benefits

for systems requiring high volume complex computations, and allowance for template-based object oriented design. Additionally, the cryptosystem is designed to benefit from C++11's low-level system optimization capabilities such as pointer operations, move semantics, memory pools, and bit shift operations. We selected Java for the prototype's web solution because of its cross-platform support, performance benefits for database access and handling of web requests and responses, and available and reliable libraries for C++ integration, database access, data transmission, web-based client/server functions, and multithreading.

The prototype's use case is based on Figure 1 and simulates a Patient submitting data to be securely stored on the PRE Server, Doctor 1 retrieving and decrypting encrypted data from the PRE Server, and Doctor 2 retrieving and decrypting re-encrypted data from the PRE Server. Figure 2 illustrates these runtime workflows. The encryption and decryption operations are performed by autonomous cryptosystems deployed in each client's web solution and the re-encryption operation is performed by the cryptosystem deployed in the PRE Server. For patient and doctor use cases, a Java servlet deployed in their respective clients encrypts or decrypts data by using a middleware implementing the Java Native Interface (JNI) framework to interface with cryptosystem APIs for encryption and decryption. The PRE Server's re-encryption use case is triggered by a request from Doctor 1 to a Java Servlet on the PRE Server that uses the JNI middleware to interface with the cryptosystem's re-encryption API. The JNI middleware referenced is discussed later in this Section.

The prototype's cryptosystem operations in Figure 2 are performed using public/private key pairs generated by each client's cryptosystem. Each client stores its key pair and transmits their public key to the Policy Server. Figure 3 shows this key management scheme. The Policy Server stores each client's public key and on request provides Doctor 2's public key to Doctor 1 for re-encryption key generation. The Patient's public key is used to encrypt plaintext data. Doctor 1's private key is used to decrypt encrypted data and is used with Doctor 2's public key to generate a re-encryption key for re-encrypting encrypted data. Doctor 1 provides the re-encryption key to the PRE Server which performs the re-encryption operation and stores the re-encrypted data as well as the encrypted data. Doctor 2's private key is used to decrypt re-encrypted data.

The prototype's Java-based components interface with its C++-based cryptosystem by way of middleware implementing the Java Native Interface (JNI) framework. For example, when a Patient enters data that needs to be encrypted and saved to the PRE Server, a Java Servlet in the Patient's web solution requests and receives the encrypted data from a C++ object in the cryptosystem. The Java Servlet then sends the encrypted data to the PRE server to store. The JNI middleware facilitates this workflow by making the cryptosystem request through a Java class that only defines signatures for native methods to interface with the cryptosystem (e.g. requestEncrypt). These native methods are implemented in an extra-cryptosystem C++ class that calls cryptosystem APIs (e.g. "Encrypt") to perform the computations for encrypting plaintext data and decrypting encrypted data. Figure 4 illustrates the JNI implementations allowing use of the cryptosystem's APIs for re-encryption (by Doctor 1 using "ReEncrypt") and decryption of the ciphertext produced by that operation (by Doctor 2 using "PREDecrypt"). The C++ middleware class is compiled together with the cryptosystem into a Dynamic Link Library file (for Windows) or a Shared Object file (for Linux). The file output by compilation is specified in CLASSPATH for the prototype's Policy Server and deployed in Apache Tomcat for the prototype's PRE Server and client web solutions.

Evaluating our architecture's prototype under the CIA triad described in Section II, the Confidentiality requirement is satisfied. As illustrated in Figure 2, this is because only encrypted data is transmitted between all client components and the PRE Server. The prototype's Confidentiality factor is enhanced by its implementation of quantum computing resistant lattice encryption techniques to compute the encrypted data transmitted between the PRE Server and its clients [6]. This benefit is not available in on-market public key security models such as RSA which have been defeated by quantum computing techniques. Additionally, as illustrated in Figure 3, the prototype does not require private keys to be shared for a data recipient to decrypt data. This benefit is not available in on-market medical data information architectures which require the sharing of decryption keys in order to share data. Thus, the prototype demonstrates that our architecture provides greater protection than current on-market security models against confidentiality-compromising attacks.

The prototype can be augmented to satisfy the CIA triad's Integrity requirement by implementing a cryptographic signature scheme within the framework illustrated in Figure 2. The prototype provides the framework for this feature because each PRE Server client generates a public/private key pair. Implementing this feature requires adding cryptographic signature computation and verification APIs to the cryptosystem. Cryptographic hashing or cryptographic checksum APIs could provide the basis for the necessary signature function APIs. The implemented cryptographic signature protocol will allow PRE Server clients to compute and send a signature element over data they transmit thereby providing the data recipient a means to verify data integrity.

The CIA triad's Availability requirement will be verified through future development and testing activities involving prototype components. The results presented in Section V were produced in the context of single users playing the prototype's use-case roles with no concurrent requests or responses for the data points in Figure 2. Scalability testing will require identifying potential bottleneck points due to the accommodation of multiple users and concurrent requests. At minimum, this will involve optimizing bandwidth use and runtime processing for (1) requests made by client components to perform the cryptographic operations in Figure 2, (2) the key distribution scheme in Figure 3, (3) the PRE Server's responses to those requests, (4) requests to and responses by the C++ and Java middleware classes in Figure 4, and (5)

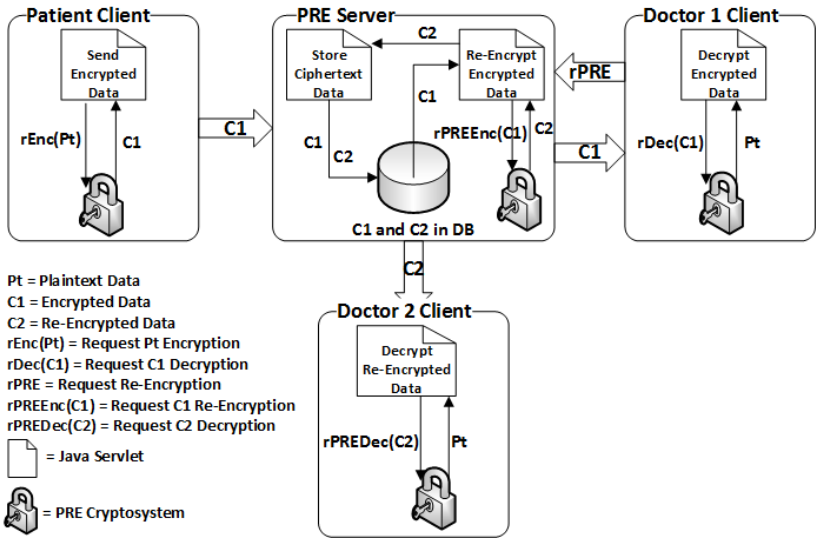

Fig. 2. Prototype PRE Server Information Architecture

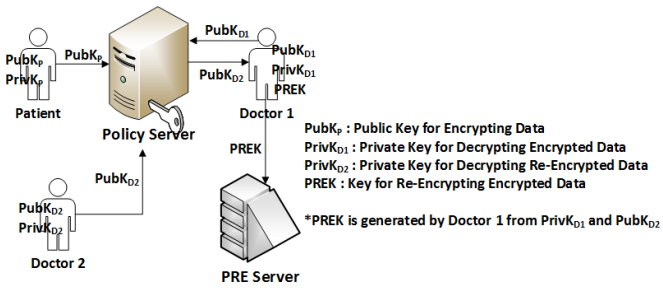

Fig. 3. Prototype Key Management Scheme

requests to and responses by the PRE Server's database storing the system's data. The resolution of these concerns is feasible through application of Client-Server system best practices (e.g. multithreading and multiprocessing) in the development and deployment of the components relevant to each concern. Additionally, concerns (2), (3), and (5) will require guarding against server-related hardware issues and denial-of-service attacks through some selection of service replication strategies. Finally, Availability satisfaction will require safeguarding against data transmission losses or interruptions for the workflows in Figures 2, 3, and 4 by implementing an effectively robust data backup and verification model. For example, a digital signature scheme could be implemented to require acknowledgements for all data transmissions and compel re-transmission of an unreceived message upon an acknowledgement failure.

V. Initial Experimentation

We performed initial experimentation for a C++ implementation of our lattice-based PRE cryptosystem using a hierarchical software architecture to enable rapid prototyping and simplify integration with embedded hardware. The design is modular and is comprised of four primary layers: (1) encoding, (2) crypto, (3) lattice, and (4) arithmetic (primitive math). Encapsulation, low inter-module coupling, high intra-module cohesion, and maximum code reuse software engineering guidelines were followed when making changes to the cryptosystem. Lattice operations were decomposed into primitive arithmetic operations on integers, vectors, and matrices that are implemented in the primitive math layer. Along with basic modular operations, this layer includes efficient low level modular mathematic algorithms. The primitive math layer provides a high level of modularity allowing the cryptosystem's user to integrate with an existing low-level library or a custom hardware-specific implementation such as a Field Programmable Gate Array.

We ran experiments on a commodity laptop and obtained runtime results of encrypting 1kb of data in 6.1ms, decrypting 1kb of data in 7.9ms, and performing access delegation operations for 1kb of data in 40ms. These results were obtained at a level of security with a work factor analogous to AES-128.

VI. Related Work

There have been previous efforts to design network security architectures for the health care system that reduce the risk of data leakage. Prime examples of these are in [10], [11]. Similar to those efforts, endpoint protection and endpoint key management are important aspects of our proposed architecture because the endpoints are the only locations where unencrypted data is accessible. Unlike those efforts, a key feature of our architecture is the use of end-to-end encryption. This feature substantially simplifies interactions between intermediate data hosts and significantly reduces the risks of data leakage due to insider attacks and compromised devices. Our architecture thus provides a more secure framework than [10], [11] because data is protected by encryption at all locations and points in time.

Prior efforts such as [12] have also addressed the secure aggregation and distribution of collected data but do so from

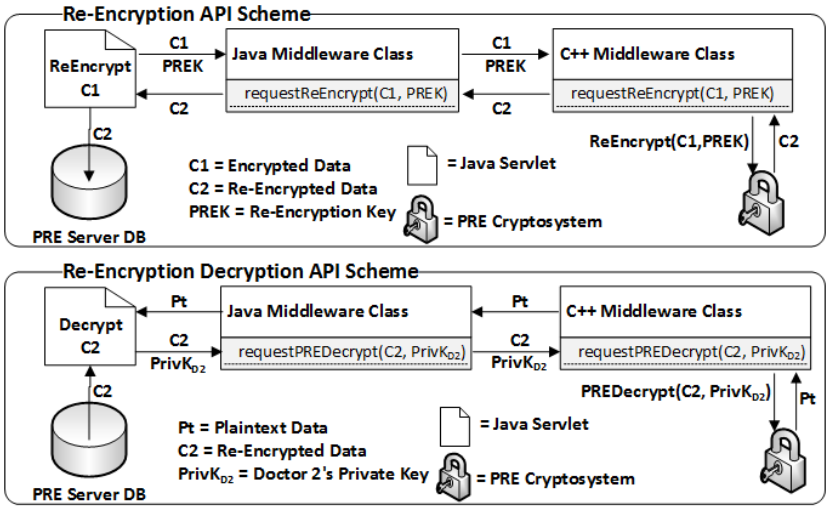

Fig. 4. Prototype JNI-Middleware Re-Encryption API Scheme

an enterprise perspective where large entities such as hospitals perform shared computations on sensitive data. Our secure information architecture focuses on patient-level data security that could support embedded mobile devices when bandwidth is an issue. However, our PRE model can also be generalized to support the secure intra-enterprise aggregation of patient level data while supporting the secure inter-enterprise aggregation of that data. For example, see [7] which provides a more general (but not experimentally evaluated) approach for data aggregation where participants perform a joint computation with results only accessible by common agreement.

VII. Conclusion

We presented a secure information architecture for delegating medical data access that protects data at all times by applying a lattice-based variant of Proxy Re-Encryption (PRE) to 1) provide end-to-end encryption and 2) prevent the need to share decryption keys. Experimental results demonstrate that our architecture addresses the deficiencies in current security models noted in Section I and the evaluation measures noted in Section II. While we focus on privacy and confidentiality concerns, our architecture can be augmented to provide Integrity and Availability protections thereby satisfying the CIA triad. For example, cryptographic signing methods could satsify Integrity and service replication could satisfy Availability against hardware issues and denial-of-service attacks.

An important benefit of our architecture's end-to-end lattice-based encryption scheme is that it enables health care entities to securely use low-cost cloud computing environments to share data while also significantly reducing vulnerability to insider attacks. For example, our architecture limits data access only to the system administrators who have decryption keys even when encrypted computing is hosted on proprietary servers. Additionally, our architecture prevents access to decrypted data until it reaches its intended recipient. The benefits of our architecture will thus reduce the operational costs of highly regulated industries such as healthcare where regulatory compliance restricts the ability to outsource data security computations to low-cost cloud computing environments.

References

[1] G. J. Annas, "HIPAA regulations-a new era of medical-record privacy?" *New England Journal of Medicine*, vol. 348, no. 15, pp. 1486–1490, 2003.

[2] R. Rauscher, "Cloud computing considerations for biomedical applications," *Healthcare Informatics, Imaging and Systems Biology, IEEE International Conference on*, vol. 0, p. 142, 2012.

[3] W. H. Maisel and T. Kohno, "Improving the security and privacy of implantable medical devices," *New England journal of medicine*, vol. 362, no. 13, p. 1164, 2010.

[4] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Advances in Cryptology-CRYPTO'97*. Springer, 1997, pp. 112–131.

[5] C. Peikert, "Public-key cryptosystems from the worst-case shortest vector problem," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 2009, pp. 333–342.

[6] D. Micciancio, "Lattice-based cryptography," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 713–715.

[7] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption," in *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. ACM, 2012, pp. 1219–1234.

[8] J. Partala, N. Keränen, M. Särestöniemi, M. Hämäläinen, J. Iinatti, T. Jämsä, J. Reponen, and T. Seppänen, "Security threats against the transmission chain of a medical health monitoring system," in *e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on*, 2013, pp. 243–248.

[9] Y. Polyakov, K. Rohloff, G. Sahu, and V. Vaikuntanathan, "Proxy re-encryption, lattice encryption, software engineering, delegating access control," Submitted.

[10] V. Gupta, M. Wurm, Y. Zhu, M. Millard, S. Fung, N. Gura, H. Eberle, and S. C. Shantz, "Sizzle: A standards-based end-to-end security architecture for the embedded internet," *Pervasive and Mobile Computing*, vol. 1, no. 4, pp. 425–445, 2005.

[11] R. Rauscher and R. Acharya, "A network security architecture to reduce the risk of data leakage for health care organizations," in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*. IEEE, 2014, pp. 231–236.

[12] A. Andersen, K. Y. Yigzaw, and R. Karlsen, "Privacy preserving health data processing," in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*. IEEE, 2014, pp. 225–230.