*MEDINFO 2010*
*C. Safran et al. (Eds.)*
*IOS Press, 2010*

*doi:10.3233/978-1-60750-588-4-869*



# Deployment of a Highly Secure Clinical Data Repository in an Insecure International Environment

**Henry Feldman[a], Shane Reti[b], Eli Kaldany[a], Charles Safran[a]**

[a]*Division of Clinical Informatics,Beth Israel Deaconess Medical Center, Harvard Medical School, Boston, MA, USA*
[b]*Department of General Practice and Primary Health Care, University of Auckland, New Zealand*

**Abstract**

*We have designed and deployed a novel approach to protecting Personal Healthcare Information in environments where a data center is remote and its physical security cannot be assured. Our "KeyServer" methodology uses a server-client-server architecture to dynamically serve keys from a distant server in a separate secure data center in the US. The approach combines pre-existing and novel techniques into a layered protective barrier around compromise of patient data. We describe how this technology provides scalable security that makes security breaches highly unlikely. With some careful planning a Clinical Data Repositories fed by Electronic Health Records can be placed in relatively insecure settings, with a high-level of security surrounding data theft, even in the event of hardware theft. Such security architecture is ideal for not only developing nations, but for the evolution of health information to cloud computing platforms.*

***Keywords:***

Security, Computer hackers, Computer data compromising, Computerized medical record system,

## Introduction

In many countries with mature healthcare IT infrastructure, security of Personal Healthcare Information (PHI) is often provided by a multilayered approach that includes robust physically secure data centers, complex network security protocols, and a legal framework with effective enforcement. These approaches may or may not be suitable in developing countries with low evolving IT infrastructure where the risks of data theft by physical and electronic intrusion can be high. A review of the literature

To address health data security issues in high-risk environments with low level technological infrastructure,[1] we designed a novel encryption and authentication pathway. We posit that our approach to protecting PHI will be generally useful for cloud-based computing solutions in healthcare.

## Methods

Data security is an important component of any electronic health record (EHR). We have designed and deployed an EHR in a Middle Eastern country that posed numerous challenges including remote data protection in a setting where we were concerned about physical security of the data. Because of low bandwidth between the Middle East and the US, deployment from a US based Tier-3 data center was not an option and we designed the system with the plan to place servers in a small commercial ISP data center located in the Middle East. Ultimately we were able to deploy in a secure government based data center, but this was well after the design phase was complete.

The core systems design consists of a MySQL 5.1 Community Edition relational database management system (Sun Microsystems) with a JAVA based application (Sun Microsystems) running on Apache Tomcat 6 (Apache Software Foundation) application server.

The original deployment strategy posed a number of problems, foremost being loss of control over the server environment. Furthermore, the database is designed for heavy secondary use of data in public health monitoring and research, making openness of the server based Clinical Data Repository (CDR) a high-risk asset.[2] Given these risks, we designed a multitier security strategy that utilizes a novel encryption and authentication pathway to provide network security and physical theft risk mitigation.

We started with the assumption that we could not assure physical security, and that our database and server code could be physically stolen given the lack of high end data facilities available in the region.[1] We were also cognizant of the fact that both the passwords and certain aspects of PHI, even with encryption, are vulnerable to preimage attacks, also known as a "dictionary attack". Passwords are notoriously susceptible to dictionary attacks,[3] and the nature of medical record numbers (MRN) as sequential non-sparse integers make them trivially susceptible to a preimage attack. Although we do have influence over password entry by users, we do not have influence over

existing medical record numbering schemes, so a secondary method of attack prevention is required.

Our initial approach was to encrypt the entire database except for any field that could be used for identification of patients. This is easily accomplished as MySQL natively supports symmetric encryption via full Advanced Encryption Standard (AES-128) support and asymmetric encryption via the Secure Hash Algorithm (SHA1). The AES-128 algorithm is a substitution permutation network encryption algorithm authorized by National Institute of Standards and Technology (NIST) for protection of information classified up to the "secret" level,[4] making it appropriate for storage of PHI. The use of AES encryption is highly secure, as no successful attack other than brute force, has been demonstrated in the literature. However, as in any password-based system, the system is vulnerable if you can discover the key since the data would be freely readable.

JAVA, while a highly secure language, has a vulnerability of being able to be easily human-readable reverse-compiled as it is not fully compiled, but rather partially compiled into byte-code. This means that storage of encryption keys in the code would be easily retrievable by having possession of the server/drives or compiled application. Having the keys in the source code also does not allow for easy changing of the codes as compilation and redeployment of the application is required.

We have created and deployed a novel "KeyServer" methodology that uses a server-client-server architecture to dynamically serve keys from a distant server in a separate secure data center in the US. The concept is that every column of every table in the remote MySQL database that needs to be encrypted, is encrypted with a column key. This key is only held in memory on the application server implementing the KeyServer client, and written to disk only on our secure KeyServer.

Each key can either be human-readable or machine-created thus protecting against dictionary attack by the use of random alphanumeric strings. Various identification attributes of the client network environment are also recorded on the KeyServer, including IP address and a hardware key of a physically separate device on the client network which is sent with every client request. These identification attributes make it very difficult to spoof the client's identity.

On boot the KeyServer client sends a getInitKey request with the identification information gathered from the client network via an encrypted web services call to the US KeyServer, and if authenticated the current keys are returned, and the application will start as shown in Figure 1.

At intervals varying between minutes and hours, the column keys on the KeyServer are changed; the KeyServer client checks its keys via a CheckKey request with the KeyServer master keys. The request sends the identification attributes described above, as well as the client's current keys. The KeyServer then checks the identification attributes, confirms the requesting client's authenticity and returns a pair of keys, the client's current key and the KeyServer's new key. For any given column, if the KeyServer and client keys are the same then the current settings are still valid and no change is made. If the client key is different to the KeyServer key then the master keys have been changed and the column is dynamically decrypted with the old key and encrypted with the new key. Figure 2 demonstrates this.

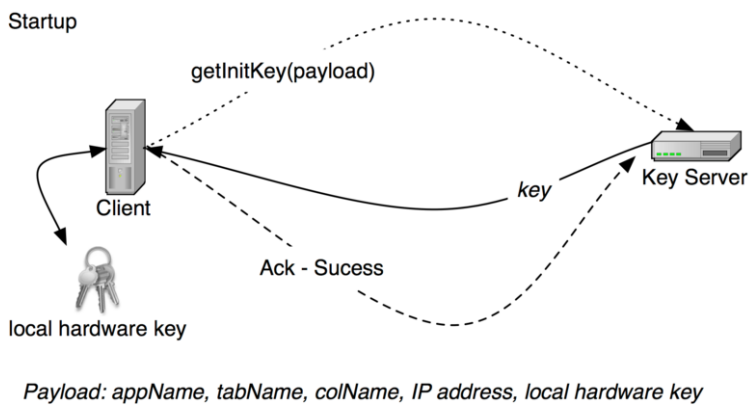


*Figure 1 - Initial getInitKey request*

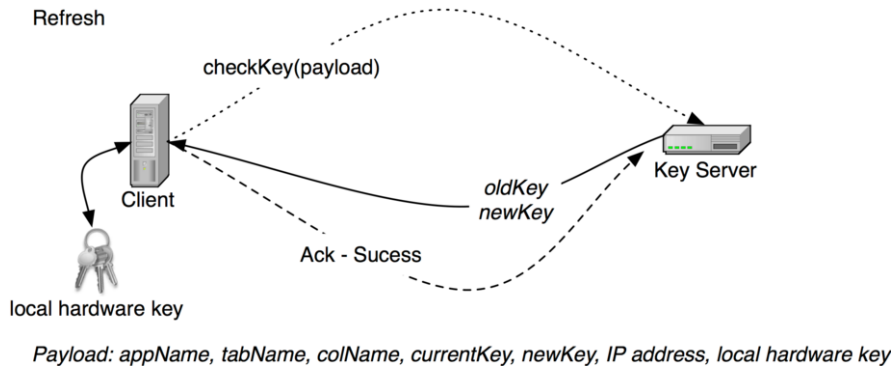


*Figure 2 - Subsequent CheckKey request*

The KeyServer never updates its own tables until an acknowledgement packet is returned from the client stating it has completed the key change successfully, as a failure could render the data unreadable. The transaction is carefully and securely logged on the server for future auditing. A network failure resets the timer, and the transaction will retry in the future.

A more detailed examination of the encryption process presents further security issues. A problem with PHI encryption is that predictable data, such as the non-sparse sequentially numbered MRN fields or names, require protection against preimage attacks. This is because if one encrypts a simple string using the same algorithm and key pair it will always produce the same hash every time. This is true regardless of the encryption method, whether symmetric or asymmetric, as both methods will produce the same output (a "collision" in encryption parlance) with the same source/key information. This is a much greater problem with asymmetric algorithms such as SHA1 where the key is fixed. It is less of a problem with random text, however patient identifiers, such as medical record numbers are simply sequential numbers 0-*n* and last names are known, so the creation of a preimage attack is trivial. For example, on a typical desktop computer you can generate the 128-bit SHA1 hash value dictionary for 10 million medical records numbers in less than 1 minute. Figure 3 demonstrates a MRN dictionary attack.

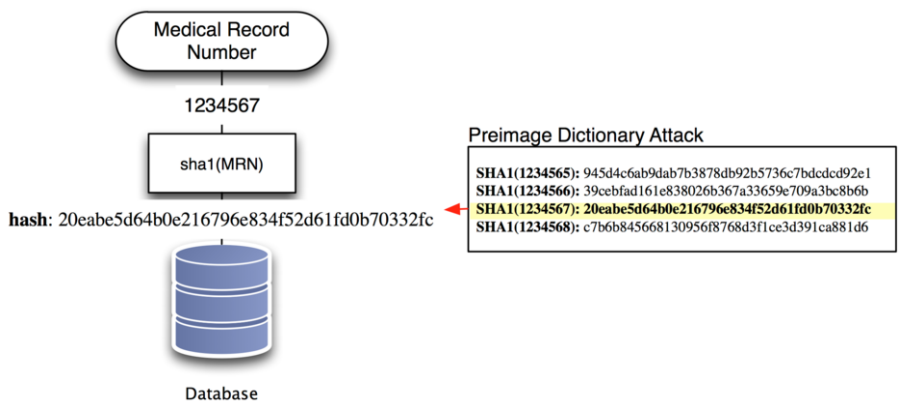


*Figure 3 - Dictionary Attack on MRN's*

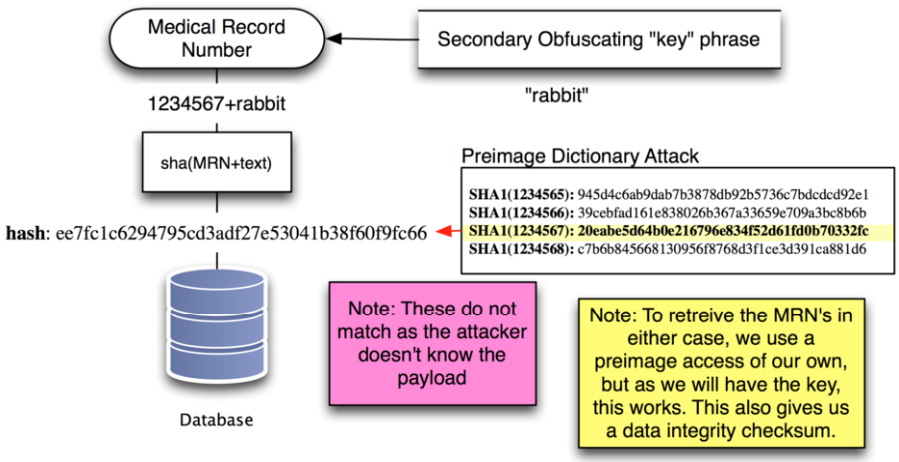


*Figure 4 - SHA1 with extra payload*

Our solution to solving this problem is to add an obfuscating payload to the medical record number. This payload key is also supplied by the KeyServer and is added to every predictable field on a per-column basis. For example, using "rabbit" as the payload key, MRN "*1234567*" becomes "*1234567rabbit*". For record retrieval we can then use our own "dictionary" knowing what the obfuscating package key was to match against. This is shown in Figure 4. If we wish to change the package key, we manually "decrypt" the column by performing our own mass dictionary attack and parsing the data back to MRN's and encrypting with the new package added. For data such as patient names, we can add this text to the start of every name, and programmatically subtract it after decryption.

While most password systems require the user to comply with various levels of onerous password change and complexity requirements, this makes users do things like write their password down next to their computers. Security breaches against passwords without access to the password file, are generally human-nature attacks such as finding the password written down or guessing familiar terms, which is not protected against by stringent password requirements. Even worse, stringent password requirements are not foolproof in themselves, and simply lower the penetration risk somewhat.[5]

However, we were additionally concerned about password file theft, which means protection against a dictionary attack. Since our passwords are stored as asymmetrical SHA1 hash values, we chose a different method. A simpler task is to take password security out of the hands of the human user, and make the file inherently highly resistant to preimage attack. Again an obfuscating payload makes a useful barrier to mass preimage attack.

Rather than forcing the user to remember these obfuscating elements, we can programmatically change their password on entry/storage on database entry to include these elements, but without access to the originating code, and knowledge of the obfuscating payload, decryption is unlikely. For instance instead of the password sent to the encryption algorithm being "*password*" it is "*passwordrabbit*" where *rabbit* is a secret key kept only within the system. The user is unaware of these changes. One downside of this, is that this package cannot be dynamically rewritten, as we have no ability to perform our own dictionary attack, since the system does not retain the passwords as entered by the user.

## Discussion

We have successfully designed and deployed a highly secure clinical data repository using our novel KeyServer technology. The technology combines pre-existing and novel techniques into a layered protective barrier around compromise of patient data. We start with network security by use of encrypted connection with RSA-certificate signed servers. We then further the identification of clients via secure session management with IP address verification and separate local hardware identification of clients. Onto this we add loading of the client side keys into memory means that if the server is stolen, on restart the local keys are destroyed, along with column level encryption. Scalable security timing of master key changing and authentication calls using system generated encryption keys make preimage attacks highly unlikely. Finally adding additional payloads to predictable data such as passwords, names and patient identifiers, makes these more resistant to preimage attacks.

There are some potential disadvantages with this configuration. Firstly, unexpected server shutdown automatically voids all keys

from memory. Depending on the reliability of server side support services, this will either be likely or unlikely. In any event, the keys can always be reloaded from the US KeyServer. Another limitation is the absolute requirement for a reliable network connection between the client and US KeyServer. Multi-column encryption and decryption processes potentially could affect overall performance, however our experience is that this is negligible.

There has been published work done that has demonstrated a cryogenic attack at retrieving keys from computer DRAM chips is possible,[6] this is a level of attack, that we choose not to protect against. This level of sophistication is outside of the level of perceived threat that PHI is likely to be subjected to. If this threat becomes more realistic, the referenced article suggests one practical countermeasure to this technique which is writing the keys to memory with large amounts of garbage data around them greatly lengthening the time required for key reconstruction, but even this is vulnerable.

With some careful planning an EHR/CDR can be placed in relatively insecure settings, with a high-level of security surrounding data theft, even in the event of hardware theft. This is made possible through advanced security practices and distributing the security apparatus across international boundaries with the primary security codes being in a highly secure data center, with no ability of the remote system to recreate its own codes. Another key to this approach is a reliable method for local client proof of identity even if stolen and booted outside of its home environment; this is the key to prevention of stolen code and database being usable on any computer not in its predefined network environment even if network hardware and soft IP addressing is spoofed. We believe these methods are generally useful for cloud-based solutions in healthcare.

## References

[1] Hamade, S. Information communication technology in Arab countries: problems and solutions. in ITNG 09 Sixth International Conference; Information Technology: New generations 2009. 2009.

[2] Safran, C., Bloomrosen M, Hammond WE, Labkoff S, Markel-Fox S, Tang P, Detmer D, Toward a national framework for the secondary use of health data: an American Medical Informatics Association white paper. J Am Med Inform Assoc, 2007. 14: p. 1-9.

[3] Cazier, J., Medlin BD, How secure is your information system ? An investigation into actual healthcare worker password practices. Perspect health Inf Manag, 2006. 3: p. 8.

[4] National Institute of Standards: Federal Information Processing Standards Publications (NIST), Announcing the Advanced Encryption Standard (AES), N.I.O. Standards, Editor. 2001.

[5] Proctor, R., Lien MC, Vu KPL, Schultz EE, Salvendy G, Improving computer security for authentication of users: influence of proactive password restrictions. Behav res Methods Instrum Comput, 2002. 34(2): p. 163-9.

[6] Halderman, A., Schoen SD, Heninger N, Clarkson W, Paul W, Calandrino JA, Feldman AJ, Appelbaum J, Felten EW. Lest we remember: cold boot attacks on encryption keys. in 2008 USENIX Security Symposium. 2008.

**Address for correspondence**

Dr Henry Feldman
1330 Beacon Street, Brookline, MA 02446
Email: hfeldman@bidmc.harvard.edu