VERIFIABLE AND REDACTABLE MEDICAL DOCUMENTS

A Thesis
Presented to
The Academic Faculty

by

Jordan Lee Brown

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2012

VERIFIABLE AND REDACTABLE MEDICAL DOCUMENTS

To Ellen and Thomas,

ACKNOWLEDGEMENTS

I would like to thank Dr. Riley and Dr. Hong for taking the time to be a part of the committee to review this thesis. And especially Dr. Blough for all of the patience, insight, and guidance he has provided since I began working with him.

I would also like to thank my friends and family, especially Ellen and Thomas, for all of the support they provide me.

Without all of your support I would not have this opportunity today.

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

This objective of the proposed research is to answer the question of how to provide verification and redactability to medical documents at a manageable computation cost to all parties involved. The approach for this solution examines the use of Merkle Hash Trees to provide the redaction and verification characteristics required. Using the Merkle Hash Tree, various Continuity of Care Documents will have their various elements extracted for storage in the signature scheme. An analysis of the approach and the various characteristics that made this approach a likely candidate for success are provided within. A description of a framework implementation and a sample application are provided to demonstrate potential uses of the system. Finally, results seen from various experiments with the framework are included to provide concrete evidence of a solution to the question which was the focus of this research.

CHAPTER I

INTRODUCTION

In the near future, electronic health information will be routinely exchanged across widely distributed entities and for a variety of purposes such as treatment, efficacy evaluation, medical research, and public health. This push is being advocated across the globe for various reasons.

In Taiwan, for example, the government requires electronic medical records for insurance purposes [30]. Domestically these electronic records may be used to increase the patient's participation in their medical routine. Various work has been done to detail the usefulness of these electronic medical records in special intrest groups. MyHealtheVet [3, 26] provides veterans with medical assistance while the work described by Weppner et al. [37] targets elderly patients with diabetes. These benefits are not limited to patient's with special characteristics. Schnipper et al. [32] take this benefit to the general public by demonstrating how patient interaction with these electronic records may benefit general medication administration. The public may also benefit from research efforts like those described by Ramakrishnan et al. [29] who propose searching these records for patterns that may be useful to medical practice. Also, studies like the work done by Kaelber and Pan [19] show the monetary benefits that may be seen in a switch to this electronic form.

However, as these documents transfer into the digital domain, they must maintain the integrity of their original source if they are to be useful in transactions. Clarke et al. describe the necessity of data verification in cardiac surgery [13], but this problem extends to all medical fields. Without this data verification these documents lose all integrity.

To date, there are no proposals for building trust into the health information exchange model, aside from direct interactions between two trusted parties. However, while such direct interactions will be common, there are likely to be many indirect interactions as well. Prime examples are: 1) the use of personal health records (PHRs) by patients to store their health information and provide it to third parties on an as-needed basis and 2) research data repositories, which gather medical data from various sources and allow multiple research projects access to different data subsets. This thesis addresses the problem of how to verify health information that is provided by an entity that was not the original source of the information. Establishing data provenance for trusted data analytics is especially problematic in distributed systems. The problem is also complicated by the need for intermediate entities to redact some of the information coming from the source, e.g. for privacy reasons.

To provide verifiability and redaction, a cryptographic primitive known as a redactable signature [18, 8] is proposed. Using a redactable signature, the source of health information, e.g. a health care provider, can sign a medical document and provide it to another party, e.g. the patient who is the subject of the document. The second party will then be able to redact information in the document that they do not want to disclose and pass the document along to other parties. An intermediate party can also merge data from different documents provided by different sources and create a new document, while maintaining the signatures of the data sources. The recipient of such a signed document can verify all of the sources of data in the document and can verify that no other party has modified the data items since they were provided by their original sources.

As a simple example of the use of such a technology, consider a situation where parents must supply proof of vaccinations of their daughter for a summer camp or school that she will attend. In this situation, the parents can receive a complete electronic vaccination record signed by their health care provider. However, certain vaccinations, e.g. the HPV vaccine, might be considered sensitive by the parents and not required for attendance. The parents would be free to remove evidence that the child received the HPV vaccine and forward on the remainder of the vaccination record to the third party. The third party can cryptographically verify that the vaccinations listed were performed by the family's health care provider without seeing the redacted information. The second party, in this example the parents, is free to retain the complete record and can provide any subset of the health information contained in it on a per-use basis. Thus, the document can be provided in different forms as many times as needed.

The objective of the proposed research is to address this question of how to provide both data verification and redactability in medical records at a manageable computation cost. This thesis details an approach that satisfies the proposed objective and implementation of a prototype implementation for this approach. Also included, are results of this implementation using sample medical records. Some of these documents are available in the public domain while others were sanitized and provided to assist with the research.

1.1 Concepts and Terminology

Below follows a discussion of the primary concepts found within this research.

1.1.1 Continuity of Care Document

The medical document specifically addressed by this research was the Continuity of Care Document (CCD) part of the HL7 [1] standards. This document is a XML standardized document which contains many sections for representing various aspects of a patient medical history. Various sample documents were used throughout the course of research; however, a sample of one such CCD, representing John Halamka may be viewed as an example. A visual sample of the document is provided in Figure 5. The sections contained within the document have both machine readable and visual portions. An example of the XML structure of the document may be found in Figure 15. Throughout this thesis references to the document imply some instance of a CCD unless otherwise stated.

1.1.2 Merkle Hash Tree

The method by which we achieve a verifiable signature on redactable data is through a structure known as a Merkle Hash Tree (MHT). For our implementation this structure comes in the form of a binary tree with three types of nodes. The leaf nodes contain health information extracted from a document as well as a hash of the item. Intermediate nodes contain only a hash value. This hash is, in general, calculated by taking the hash value of the nodes two children, concatenating those hashes, and hashing the concatenated value. The root contains a hash that is calculated the same way initially, but then is signed by the information provider to insure verifiable data.

These tree structures may contain only one layer or multiple layers of signatures. A tree with multiple levels of signatures may have signed root nodes dispersed throughout the tree. To construct a tree with this characteristic, a set of preexisting single-level trees are inserted into another tree for signing. The roots of those trees may be seen as the leaves to the new tree. A sample of this structure may be seen in Figure 2(a).

From this point, any mention of a tree structure or node make reference to a MHT unless specified.

1.2 Use Cases

The work described in this thesis has three primary use cases. The first is a patient centric use case which would provide the patient with real-time control over medical disclosures. The second scenario addresses the possible benefits in the research field. The last provides a clear documentation over medical services provided by various health care providers. It should be noted that these scenarios are not mutually exclusive. Meaning, a realization of an application for one of these scenarios does not prohibit applications for other scenarios. In fact, if handled correctly, the same structure could be used to address all three situations.

1.2.1 Patient Centric

In this use case, the objective is to provide the patient with the ability to provide medical credentials in real time. To accomplish this, the proposed work flow is as follows.

A patient may visit a various number of health care providers over the course of time. Due to such an occurrence each provider is likely to have a number of medical claims to add to the patient's medical history. Each health care provider that produces medical data relevant to the patient could provide this data in the form of a CCD. In the process of producing a new CCD the data items could be kept for storage within a MHT. That is to say, the machine readable portions of the document could be inserted into the document as well as stored in a list for the creation of a MHT. The benefit to parallel creation of the two items is time. If the CCD is created prior to the MHT without saving the health information outside of the document the health items must be extracted from the document which is a very costly operation. The data extraction process will be detailed later in this thesis, but at a high level uses XPath queries which can be very slow at times. This extraction may be avoided if the items are also stored outside the document upon creation. Once the tree is constructed the provider would calculate the hashes and sign the MHT root. This structure could then be provided to the patient along with the related CCD in a secure manner.

The patient could store each structure individually, replacing structures if a new one was provided by a given source. The alternative to individual storage would be a global storage in which a global tree is created with each of the sub-trees from the providers included within the branches of the global tree. The positive side to such storage would be that the patient does not need to construct a tree for every transaction. But, such storage would require reconstruction upon any addition or alteration in data. Also there may be times at which entire sub-trees are not needed which would require large redaction sets. For these reasons it is suggested that the items be stored individually.

Upon the need for a medical transaction the patient could generate a global tree by combining only the sub-trees that contain information required by the current transaction. This global tree could then be signed by the patient or a third party to account for the given collection of sub-trees. They could then redact any information not needed and provide the signed structure as well as a sample CCD as credentials for the transaction.

1.2.2 Research Distribution

A, currently, more common use case of distributing redactable medical records is that of medical research. When collecting data for medical research it is often the case that these records must be reduced to contain the minimum required data needed for the task at hand. There is a process of approval for obtaining the records as well as the sections of these records that may be obtained. There is also typically a hierarchy of access. Specifically, a research head might be allowed to see a larger sub-set of the data than the assistants working under that supervisor. This produces the need for multiple levels of redaction in this research setting while maintaining the verifiable nature of the data.

In this scenario, the data could be globally compiled into a central repository. The meaning of "globally" here depends on the set of contributing sources to the health information. This could range from a few medical establishments who share medical records for common patients to a truly global scale. Each patient could have an associated global view of represented data as well as the MHT structure. This would consist of sub-trees for each contributing source of data that may be updated or removed when needed. This data could be queried for general information or for specific health records by approved users of the system. Then, through use of a preexisting release policy, the data could be returned in this MHT structure with the appropriate elements already redacted. An accompanying CCD could be provided here as well. Then as data was distributed among the research group, appropriate future redactions could be handled by the supervisors based on the required disclosure policy.

1.2.3 Data Provenance

The third use case uses the signed information as a form of record keeping. The work by Cadenhead et al. [12] demonstrates a method for providing provenance with the ability to redact certain steps in the data trail. However, with this signature approach a level of provenance is inherent. Through use of signatures, records detailing where medical services and results originated may be maintained and used for review processes or auditing. The signatures would likely come from physicians providing the data as well as signatures from those who must review the data. To maintain such a structure, information could be generated and signed by individuals at a medical institution. Based on the desired structure of these records any individual physician may have multiple trees that they produce depending on the review of the data that must follow. For example, a patient's lab results and imaging results might need to be reviewed by different people at a higher level so the doctor could produce one tree for imaging and one tree for lab results. These trees would then be compiled into a larger tree and signed by the reviewer of each section representing the review processes of the given establishment. These records could then be kept until needed to demonstrate verifiable information from the doctor's or establishment's records.

CHAPTER II

RELATED WORK

The medical field is rapidly moving from the world of paper to the electronic domain. Applications such as personal health record repositories and standards such as NHIN Direct are leading the way for this kind of transition by allowing patients and doctors access to electronic medical forms and results.

Microsoft HealthVault [2] is among the most well known personal health record repositories. It has a large infrastructure in place to allow for the storing of patient medical records. However, these repositories are still fairly new applications and there are many downsides to these services currently. One of their disadvantages is that the data cannot currently be verified for validity and integrity. This creates a problem when the information is to be distributed, because consumers of the information can not fully trust it and therefore cannot use it for critical purposes.

The work done in this implementation is an application and evaluation of the approach discussed in the paper by Bauer, Blough and Cash [7], but in the context of medical documents. The Bauer, et al. paper stems from the work by Johnson et al. [18], which has been a starting point for many works in the area including an efficiency increase through RSA use by Lim and Lee [21]. Johnson et al. discuss in detail the concept of using a MHT, described initially by Merkle [22, 23], to allow a set of items to undergo verifiable redaction by the user. Bauer et al. expand this work to include trees containing data from multiple authorities in which each authority signs a sub-tree of its data. This functionality is the basis for this design and a description is included herein; however, more information may be found in [7]. Since a single patient's medical record might contain information from multiple health care providers, we need the ability to include signatures from multiple authorities in the MHT structure.

This thesis looks to evaluate the suitability of basic redactable signatures for use with medical documents. The capabilities of interest are basic verifiability and redaction, and the performance of the associated operations in the medical context. This concept has been examined in very similar terms through various sources. Wu et al. [38] provide a web based approach to medical document redaction. However, they provide redactability through a concatenation of every redactable block of the document opposed to the MHT approach. They also do not provide details of the granularity provided by their method or computation times required by their system. The respective works by Slamanig et al. [34, 33] handle each node in the XML document as a node within the MHT that may be redacted.

In taking this work further, it would be possible to incorporate different enhancements that have been proposed for redactable signature schemes. For example, Bauer, Blough, and Mohan show how to encode data dependencies within a MHT [8], which allows the data sources to enforce simple policies on how their data can be disclosed. The work by Miyazaki, et al. [25, 24], uses a commit vector and bilinear maps respectively to allow for data sanitizing, which is their term for redaction. Their work provides methods for hiding the number of items removed as well as controlling when items may no longer be removed from the document. Haber, et al. [15], discuss the use of hash trees to allow for data redaction and data dependency trees in the context of data generalization and pseudonymization. Izu, et al. [16], describe a system, which they name PIATS, for tracking redactions in a way that does not prevent redactions from being made by an untrusted source but allows viewers of the data to see who redacted what data. Other characteristics that have been analyzed in these signatures include message replacement through use of chameleon hashes [6], transparency of redactions [9], aggregating signatures [17], and message structure or ordering removal [31]. Some of these characteristics, particularly signature aggregation and structure order hiding, are found in some form in the work presented. An implementation and comparison of some of these signature schemes may be found in the work by Pöhls et al. [27].

Other work has been done in the area of data integrity, which can be considered relevant to the work described here. The work done by Polivy and Tamassia [28] describes a process for using XML signatures in conjunction with Web Services on sets of data to provide easily distributed responses to queries on distributed authenticated dictionaries. Since the XML signatures and Web Services follow a set of common standards, the data is easily transferred from source to client and easily verified when obtained. This method could be used to send health information but does not account for the need to redact information from previously signed data. Bull, et al., take "content extraction signatures" from [35], which details multiple schemes for achieving redaction for a given source, and expand them to work with XML signature standards in both "single dimensional" [11] and "multidimensional" [10] scenarios. Here, single dimensional refers to treating each data item equally in redaction selection and multidimensional refers to grouping items that have dependencies on a key item such that those items may be present or removed only if the key item is disclosed and default to hidden if the key item is hidden.

This work does not explicitly consider data confidentiality, typically achieved by encryption. However, redactable signature schemes can easily be integrated with any encryption scheme or a secure storage architecture to operate within a given privacy framework [20]. Notable when discussing encryption for health information exchange is NHIN Direct [4], which is an emerging set of standards for secure health transactions. However, the standards deal only with a single verifiable source sending data to another entity. The standard provides for encryption together with a digital signature on the entire set of data items. This does not allow the recipient to further pass along only specific items from the set and maintain source verifiability. The work done here addresses this by providing the patient with a dynamic set of health items, signed by medical authorities, which a patient may distribute selectively. This could be incorporated into NHIN Direct by expanding the standard. It could also fit within other encryption schemes [36] or privacy architectures [14, 5].

CHAPTER III

DISCUSSION OF APPROACH

Below, a detailed look into the concepts behind the entire process of creating and using these verifiable and redactable medical documents will be given. As mentioned earlier this discussion falls in two primary categories. The first will be an in-depth look at the inner-workings of the MHT. While the second will describe the CCD.

3.1 Tree Structure

This first section will describe in detail the Merkle Hash Tree. This structure may be seen, at a high level, as using a binary tree of hash values with a signature at the top to cryptographically verify arbitrary sub-sets of data. Below, a description of the nodes within the structure, possible interactions with the structure, and extensions on the implemented structure will be given. Throughout the description the nodes referenced may be visualized with the help of Figure 1.

3.1.1 Node Description

3.1.1.1 Leaf Nodes

To form this tree structure we begin construction at the leaves of the tree. For our application we randomize the items stored within the tree to prevent information leakage through examination of which elements have been redacted. For example, if the items were stored in order of extraction and an encounter was redacted which had another encounter listing before it and after it, someone would be able to infer that an encounter was missing at least. Other information such as a range of dates or even the encounter location might also be inferred from such an attack as well.

Each leaf of the tree, during construction, is given one of these items for storage.

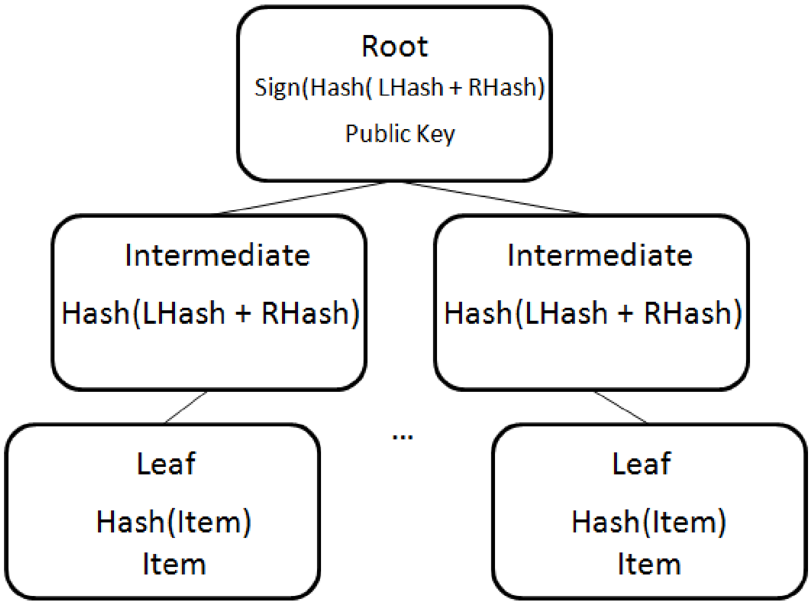


Figure 1: Tree Structure

In our implementation each element is stored as a string of characters, but this could be expanded to work with binary items. Each item is then hashed and the hash value is also stored in the same leaf node. This storage of the items within the leaves is not necessary. It is only the provided implementation. These items may be stored elsewhere with a pointer stored within the node to the data item itself.

3.1.1.2 Intermediate Nodes

Every node that is not a leaf, in this implementation, contains a hash value without an accompanying item. The hash value for these nodes is calculated by a concatenation of the two child hash values which is then hashed again. This process works from the assumption of a collision free hash. This assumption states two things. Firstly, that given random inputs to the hash function the hash values should be equally distributed to avoid two random inputs matching in output. Secondly, it should be hard to provide any input that has the same hash value as another input. From this assumption, it should be improbable to find any other two hash values that form this same hash when concatenated. This assumption will propagate up the tree to the root.

3.1.1.3 Root Nodes

The root nodes of a given tree have a hash value that has been calculated in the same fashion as an intermediate node but has also been signed by the data source. Based on the collision free hash assumption made previously, the signed hash provided in the root should only be obtainable with the two children used to calculate the hash.

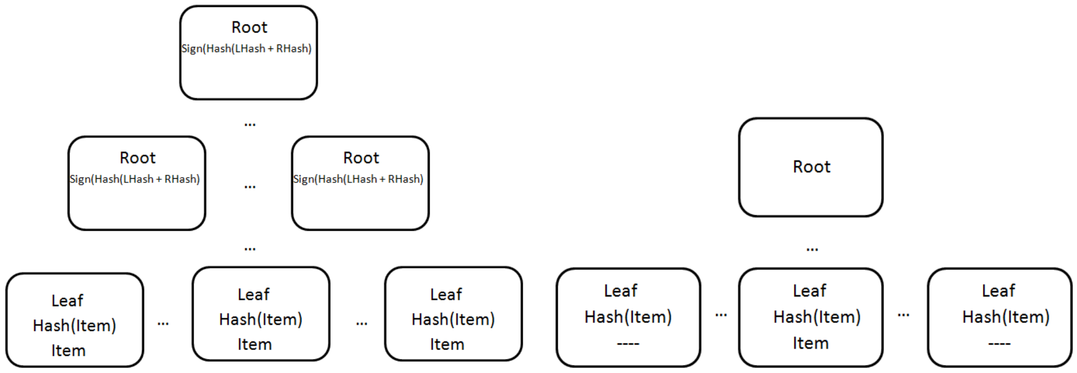
The term root node may be misleading. There may be sub-trees for various sources contained within a larger global tree. This may be seen, for example, in the case where a patient has multiple trees from different medical providers which they wish to disclose for a given transaction. Each sub-tree would have its own root node signed by the provider and the global tree would likely be signed by a patient or a third party distributor. These sub-trees may be extended to the desired number of verification steps required in a given transaction. This may be relevant in the data provenance use case where various levels of data review are required before the data is valid.

3.1.2 Interaction

There are three primary interactions that make take place within this implementation. A person may wish to create, redact, or verify this MHT. This section will provide a detailed examination of these interactions.

3.1.2.1 Creation

To create a MHT, the data elements to be signed must first be provided and stored in leaf nodes. These items are hashed and the hash value is also stored within the leaf node. All remaining nodes then calculate their hash values through hashing a concatenation of their two children node's hash values. Finally, the root signature is signed by the source of the data so that it may be later verified.



(a) Multi-level Tree (b) Redaction

Figure 2: Tree Samples

A sample tree with sub-trees may be examined in Figure 2(a). If a tree is being constructed from existing sub-trees the construction is similar to that mentioned earlier. Each tree is treated as a new leaf item. None of the hash values below the sub-tree's root signature will be or can be changed so only the root items of these sub-trees must be handled. The hash values are not calculated for these root nodes because they already have a signed hash in place. The nodes between these signed roots and our new global root must now calculate their hash values and the global root must now be signed.

3.1.2.2 Redaction

To redact an element from the tree, the element must first be found. After searching the tree and finding the data, the item is removed from the node leaving only the hash value. A visualization of this removal may be seen in Figure 2(b). Each parent node above that value then checks to see if it has any remaining leaf data beneath it. If no data is contained beneath that node all nodes below it are erased and it becomes the new leaf node while keeping its existing hash value. This process prunes the tree to save on memory space as well as future search times.

3.1.2.3 Verification

Verification is a key function in this implementation. To demonstrate how this function works the following description is provided:

For each leaf of the tree a check is performed to see if an item is present or not. If no item is present the hash value stored in that leaf is accepted as valid. It is acceptable to trust this hash because if it is not valid the calculation of other hashes within the tree will fail, based on the collision free hash assumption, and the whole tree will be rejected. If an item is present within the leaf its hash value is calculated and compared to the one stored in the node. Then for the rest of the nodes in the tree their hashes are calculated again as they were upon creation, concatenation and hash of the children nodes, and if any of these hash values do not match what is provided the verification fails. Once the root node hash value has been calculated, the signature is verified by comparing the signed hash value with the expected one. If there are multiple sub-trees, the process continues verifying all intermediate and root nodes until the global root has been verified.

If verification fails at a given point the safest thing to do is reject the entire tree. However, there may still be some data that is verified despite this failure. For example, if verifying a tree with two signature levels and the failure occurs between the global root and the sub-tree roots. Each of the sub-trees has been verified at that point to come from the provided source and the data beneath has passed verification. Therefore, it may follow that for some cases the verified data may be accepted. However, in the provided implementation no assumptions of that nature are made and the entire tree is rejected upon any verification failure.

3.1.3 Extensions

Throughout the course of this research additions or alterations to the standard MHT were considered and may prove to be useful in future implementations. An analysis of these findings will be provided here.

3.1.3.1 Data Dependencies

As seen in the work done by Bauer et al. [8] it is possible to enforce a release policy on the data to prevent out-of-context information disclosure. This is done at a high level through a creation of a graph structure using hashes in a similar fashion to the MHT to enforce data dependencies. These graphs consist of "AND" and "OR" dependencies. For example, the policy may be such that A should only be released if B AND C are released. Another example could be, A may be released so long as B OR C is released with it.

This structure could prove useful in placing medical documents in this tree structure by allowing a higher level of granularity for the medical items while enforcing a disclosure policy which would maintain the information structure intended by the data provider. For example, the results section of the document often stores a large number of test results in a single entry. However, it may be the case that some of those elements could be hidden for a given transaction but others are needed. By storing the entire entry as a single element within the tree this is not possible because the entry may either be removed or kept as a whole. However, using data dependencies the shell of the XML entry may be maintained as dependent upon any one of the included results. By using this method the entire entry or a sub-set of the entry may be released. Such functionality was implemented, although the implementation was not directly applied to a section of data within the CCD.

Such an addition would add a level of complexity to visualizing the redaction functionality of this structure due to the fact that this dependency graph is stored entirely within a leaf node. A user of this system would have to be given an option to remove a small element of the entry or the entire entry.

3.1.3.2 Information Placement

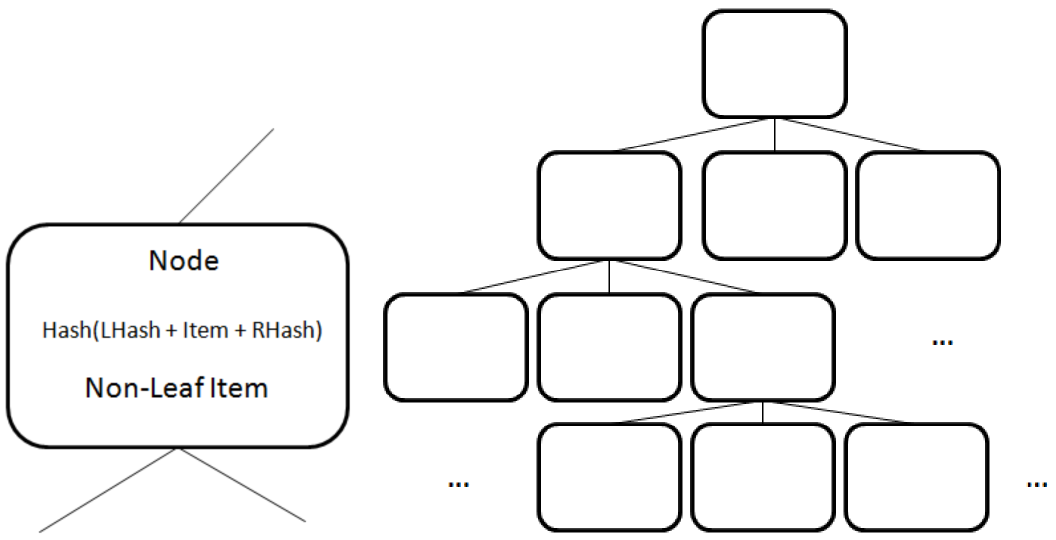
Another possible addition to this work could focus on placing key information throughout the tree structure instead of only at the leaves. A sample of this type of structure may be seen in Figure 3(a). This addition would also serve as a form of data dependency by forcing that item to be released if any of the items beneath it were also released. A motivating example would be placing some sort of alias which would allow an indirect identification of the patient to be stored in the root node. This would insure that under a set of regulated circumstances someone could verify that the information provided was representing a given patient.

To implement this, each node could have the addition of a space to store data along with its hash. The new hash would be calculated from a concatenation of the leaf nodes as well as the hash of the stored data. Thus, the resulting hash and all hashes at higher levels would depend on that data.

The example of results data mentioned earlier could also be handled in this method. Using this method for handling results data would require all elements of a given entry to be grouped within the tree which is something that would need to be done at the time of tree creation. However, such an alteration would leak information during redaction. When similar items must be grouped together an adversary could determine at a minimum the type of item that has been redacted in those situations.

3.1.3.3 Number of Children

As the use of a method such as this becomes common in medical transactions, it is possible that the number of transactions will increase greatly while the number of items used in a given transaction may begin to decline. In such a situation it may be cumbersome to send the large number of hashes that represent the path to construct the root. An interesting future work could include a look into changing the number of child nodes each node has. Doing this would allow examination of the changes in



(a) Information Placement (b) Number of Children

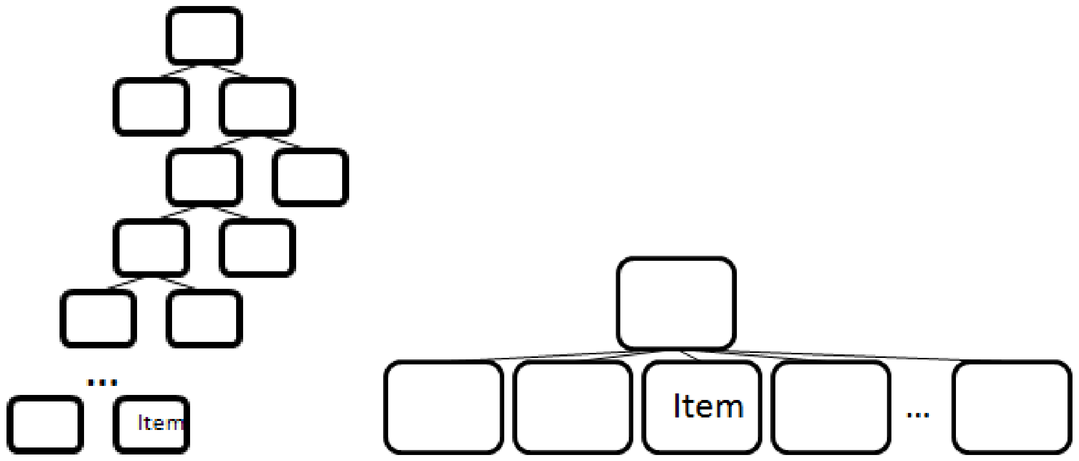
Figure 3: Tree Extensions

the amount of data that would need to be sent to represent a transaction with small number of disclosures. In Figure 3(b), the tree has been extended to three children per node.

When examining this trade off it is easiest to examine the cases on either extreme. For example, consider the case of a transaction which starts with an element count of 1024 and two tree setups. The first setup will match the implementation of a binary MHT. The second is a tree with an arbitrary number of children nodes. This allows for a root signature with 1024 children.

First consider the case of sending all of the items. Neither tree size may shrink and all of the data must be sent. In the binary tree, there will be data for 2047 nodes sent in the transaction. This is because a full binary tree with N leaves contains (2N-1) nodes. However, in the other tree only data for 1025 nodes (1024 leaves and a root) will be sent which is much more efficient.

Now consider the cost of sending one element. In the tree with two levels, seen in Figure 4(b), none of the leaves may be removed because the root hash depends on all children hashes. However, in the binary tree, Figure 4(a), there is only a single path



(a) Binary Tree (b) N Tree

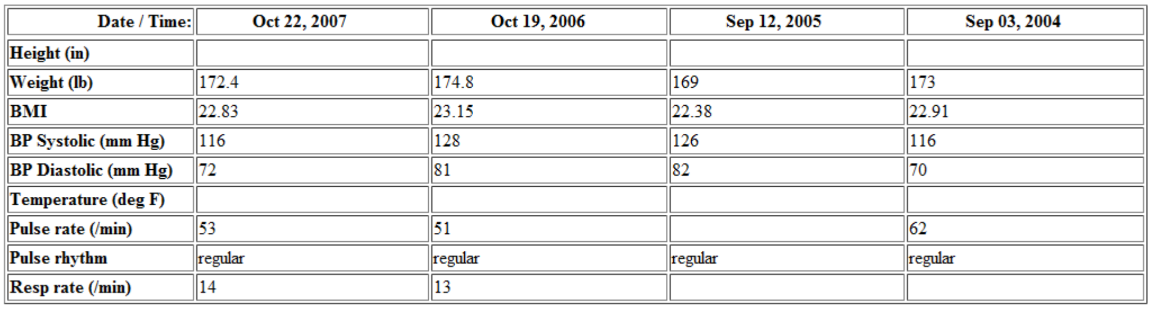
Figure 4: Disclose Single Element

that must be kept. With a binary tree of 1024 leaves there are 11 levels of nodes. At each level we must send both children of the node we need to verify. This totals to only 21 nodes which is much more efficient.

Much overhead could possibly be saved by examining the usage patterns of these structures in a real world implementation. Through analysis of the average count of elements per document and number of these elements used per transaction, a more efficient structure may arise somewhere between an arbitrary child count and two children.

3.2 Document Handling

The XML structure of the document allows for items to be uniquely stored and placed back within an empty document. A detailed look at the data extraction and re-population process, as well as, how these operations insure that the meaning is retained across arbitrary redactions is provided within this section. Samples of the visual representation, Figure 5, and the XML representation, Figure 15, for the Vitals section of the document have been provided to assist with the discussion.



Date / Time:	Oct 22, 2007	Oct 19, 2006	Sep 12, 2005	Sep 03, 2004
Height (in)				
Weight (lb)	172.4	174.8	169	173
BMI	22.83	23.15	22.38	22.91
BP Systolic (mm Hg)	116	128	126	116
BP Diastolic (mm Hg)	72	81	82	70
Temperature (deg F)				
Pulse rate (/min)	53	51		62
Pulse rhythm	regular	regular	regular	regular
Resp rate (/min)	14	13		

Figure 5: Vitals

3.2.1 Data Extraction

The extraction process is used to gather data for the tree structure. There are only two primary cases for which items need to be extracted from a standard CCD. The first type of item is header data. This information does not fall in a given section but rather contains the high level information relevant to the patient. Information such as name, date of birth, contact methods, and primary health care provider may be found here. All of this information is found at a high level within the XML structure of the document so each of these items are extracted and tagged with a string marking them as header data. The second type of data is found within the various document sections. Each of these items is contained in an individual section entry. To extract this data, each section is searched and all entries from that section are removed and tagged with the section template number. Each section contains entries of a particular template. Some sections contain data that has a different meaning but may be represented in a similar fashion. The template numbers on these entries are the same and therefore, the data requires a section number tag to insure proper return to the original section.

The visual tables and the remaining structure of the document itself are ignored. The visual portions of the document must be ignored because they are not standardized. The document contains these portions in order to efficiently convey information to the reader about the information stored in the machine readable portions. This is done differently across document implementations. Because of this, there is not a reliable way to extract this data in a redactable fashion. Finally, because the data cannot be extracted and separated efficiently, there is no method for using this data to reconstruct the visual portions. Therefore, these portions of the document are ignored when extracting data. They may be reconstructed for any given transaction based on the information disclosed on that instance. Because the visual portions are ignored it is required that all important information is included in the machine readable portion.

3.2.2 Document Re-population

Upon verification the document must be repopulated with the health items that were provided with the tree structure. To do this a skeleton document with no present items is used as a base. All of the health information provided by the tree is examined and placed back into the document in the section from which it originated. This is done with the tags added during data extraction. Because the visual tables were not extracted in the previous section these items must be constructed from the present items. To do this, the XML standards of the CCD were examined and the relevant data from each entry is collected for visual representation. Based on the information collected, appropriate tables are constructed which provide the reader with an intuitive layout for representing the data. Because each section contains different entry types the extraction and table construction are handled on a case-by-case basis. Due to the possibility of oversight when re-constructing these portions of the document it is possible that some represented data may not be given visually in this implementation. But any data stored within a disclosed entry of the document will remain in the machine readable portion.

3.2.3 Document Integrity

During this process of redaction and document reconstruction, care must be taken to insure that the meaning of the original health information is retained regardless of any action taken by the distributor of the medical items. To guarantee that all items will maintain their intended meaning let us examine the possible changes that can be made from creation to verification.

The assumption of the collision free hash has already been mentioned, but is the primary foundation for insuring the integrity of this structure. This assumption asserts that it should be hard in polynomial time to find two inputs to a hash function that return the same hash value. With that said it should be hard to either construct another data item with a hash that matches an existing one or to find two new hash values that when concatenated produce a hash value already on the tree. This insures that the tree will not verify with arbitrary changes made in the leaves. Because, to make a change in the leaf, one of the two collisions mentioned earlier must occur.

With that assumption, the only action possible between the time at which the tree structure is created to when it is verified and the new document is constructed is redaction. Any person with access to the tree may redact an item, but due to the method by which information is collected and stored all items in this implementation are independent from one another and self contained. Also because each item is tagged with the section from which it came before it is added to the tree structure there should be no method for saying that information originating from one section belongs in another section. For example, someone may try to forge medical data by saying that information found in the problems section, which details the patient's medical complications, is actually attributed to their father and belongs in the family history section. The tagging of each item prevents such a forgery because the tag becomes a part of the signed element and may not be altered.

CHAPTER IV

IMPLEMENTATIONS

Throughout the course of this work, various software artifacts were produced to demonstrate effectiveness or possible uses of the system created. The description of these implementations is provided in two sections. The first section covers the development of a framework that represents the MHT approach as well as the data extraction and re-population of the CCD. The second demonstrates an application that was developed to interface with the framework to run on Android devices. This system has been tested through emulation only as a proof of concept. The framework implemented should be platform indifferent and because the implementation was completed in Java this characteristic should hold true. Initial tests have demonstrated the ability to generate these signatures on Windows and verify them on Linux. However, more tests would be required before this implementation is placed in a real application.

4.1 Framework

The framework of the system consists of the functionality to interface with the CCD through XPath queries as well as the MHT through function calls. To keep in line with the logical flow of the system, description will begin with the data extraction. To extract the data, a document file (.xml) must be given to the system. Given that document, XPath queries are performed to extract and tag the relevant XML nodes, from within the document, for storage within the MHT.

These nodes consist of two classifications of elements. The first may be considered header information. These provide information about the patient such as name, address, and contact information. In the current implementation, these nodes are

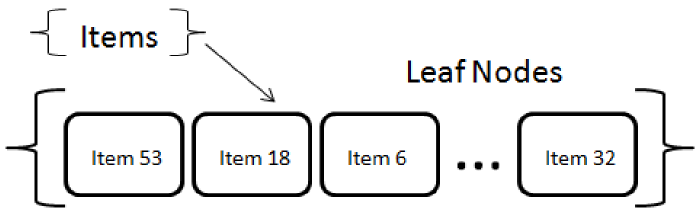
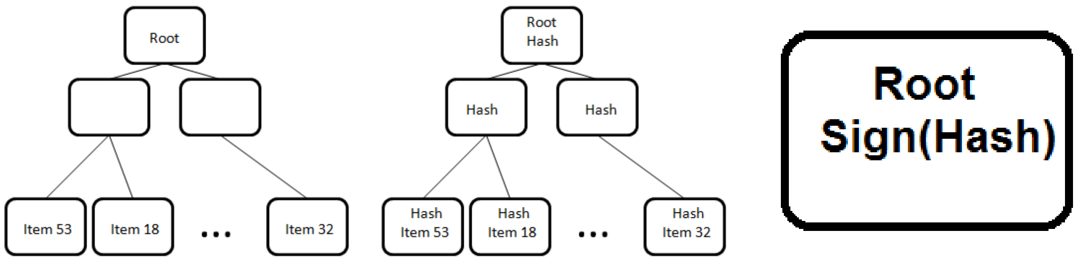


Figure 6: Create Leaves

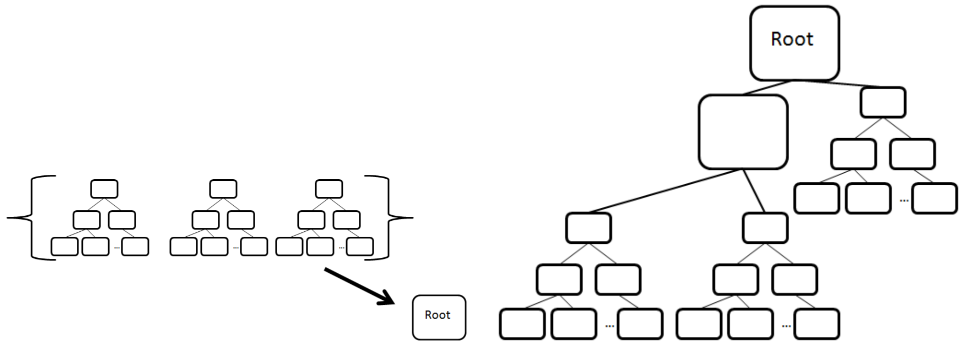
not classified but just taken for storage in the tree. The order of placement upon reconstruction will be arbitrary, but this does not change the meaning because no order of these nodes is imposed on the CCD. These header nodes are tagged with "header" at the beginning to insure they are replaced correctly. The second set of nodes originate from the various sections within the document. They are the machine readable portions containing the relevant medical information of the patient. These nodes are stored under the name "entry" in each section. When extracted these are tagged with the template identifier of the respective section so that all entries may be returned to their original section upon reconstruction.

To create a MHT from this data, the leaf nodes must be constructed first. The items are not automatically randomized so this must be done when constructing the entire tree. For this implementation, a random element is pulled from a list when first constructing a leaf. This process of leaf creation is seen in Figure 6. These leaves are then passed to a root node structure and the basic structure is formed (Figure 7(a)). A call to create the appropriate hash values must be made next during construction. This is done through a recursive call from the root. As the root attempts to calculate a hash value it needs the hash values of the two children nodes. This stops at the leaf nodes which calculate their hash values based on the information contained in the node. A sample tree with calculated hashes may be examined in Figure 7(b). And finally a public and private RSA key must be provided to sign the root node, seen in Figure 7(c). The structure was made in such a way that it may be serialized and saved for data transportation.



(a) Add Leaves (b) Calculate Hashes (c) Sign

Figure 7: Tree Construction



(a) Sub-Trees (b) Global Tree

Figure 8: Creation With Sub-Trees

The public key is only required in this implementation for verification. In this implementation the public key has been stored in the node . This method works well as a proof of concept but in a final implementation the node would likely only contain a link to the source's public certificate. By using only the public key we have no method for verification of who actually signed the data. However, with the public certificate stored or a link of some sort to the public certificate the source of the data may be verified.

If constructing a tree from sub-trees, the root node of each tree must be passed to the global root node. This process is shown in Figure 8. A call to create the hashes and sign the root is similarly needed in this case as well. This is a recursive call as mentioned earlier but does not go to the leaves of the tree. Once this call reaches a signed root the recursion stops and all nodes may then calculate their hash values. A sample tree with three sub-trees may be seen in Figure 8(b).

To redact data from the tree the element to be redacted must be provided. After each redaction the system will automatically try to reduce the size of the tree. For this implementation, the method for providing the item to redact has been setup as follows.

A list of items is collected from the tree. From this list any items to be removed from the transaction are provided back to the root node. A recursive call is then made to remove the requested information. If the information is found and removed, notification is returned through the root node. This redaction does not take into account the possibility of multiple occurrences of the same item. While possible, accounting for that possibility would only increase the time required for the common case by extending the search to the whole tree every time. The multiple instances may still be accounted for by redacting the element for every time it is included in the tree. This case may be possible if multiple data sources include identical information such as name or contact information.

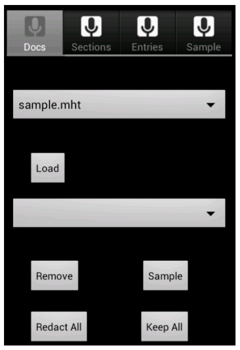
Because the public key is stored with the data, for this implementation, verification is quick and simple. Each node in the tree is compared with the expected value from the given items. The root is then verified using the provided public key and this takes place across all lower levels of trees for the given verification request. However, with this method of verification there is no link to the origin of the data and was only used in this implementation to simplify the signature and verification portions. This problem is easily addressed by storing a public certificate or a link to a public certificate insuring that the data has a verifiable source. Verification using the public certificate would be similar, only distinguished by the fact that the public key must be gathered from the certificate.

To re-create a document from a provided set of claims a skeleton document is provided. Using the tags provided with each element the data is placed back into position. Then using the set of standards provided for CCDs, each element is processed by the appropriate section handler to extract the individual information provided in each data element. This extracted data is used to create visual tables and representations for the viewer of this document. For example, the elements in the advance directives section may possibly contain information about the directive, witness, effective dates, and reference material. However, if all of the elements repopulating this section contain only a directive and effective dates at most, a visual table will be constructed using only those two fields. These tables are also placed in their respective sections and the final document may be saved as an XML file for transportation or viewing.

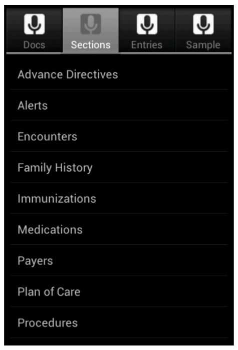
4.2 Mobile Application

To demonstrate the scenario of patient control over medical document disclosure, an application was developed using the above framework to provide a sample user interface to the redaction portion of the system. The redaction portion was selected for this implementation due to the fact that the other portions of the process may be almost completely automated. For a medical establishment providing copies of these documents, the document to create a MHT from would need to be specified as well as information specifying the doctor signing the data. At which point the system could easily extract relevant information and form the tree and sign the root with the appropriate signature. During a transaction, upon specifying the tree to verify, a system could easily validate the tree and re-populate a document with the provided information. There may be scenarios where the redaction may be defined as a policy and automatically carried out, but in the patient controlled scenario it is likely that the patient would make these redaction decisions per transaction and would require a method to interface with the system.

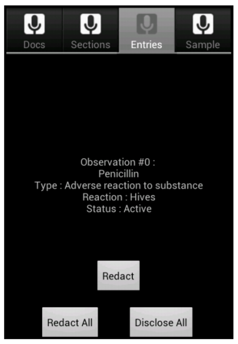
The redaction process is complicated and would need to be made intuitive for such a system to become useful. The application was developed with different levels



(a) Documents

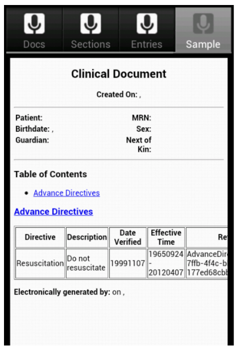


(b) Sections



Observation #0 :
Penicillin
Type : Adverse reaction to substance
Reaction : Hives
Status : Active

(c) Entries



Clinical Document

Created On: ,

Patient:	MRN:
Birthdate: ,	Sex:
Guardian:	Next of Kin:

Table of Contents

- Advance Directives

Advance Directives

Directive	Description	Date Verified	Effective Time	Re
Resuscitation	Do not resuscitate	19991107	19650924 - 20120407	AdvanceDir 7ffb-4f4c-b 177ed68cbb

Electronically generated by: on ,

(d) Sample

Figure 9: Application Layout

of interaction in mind. At the global layer the users have control of the documents to include and the ability to begin the redaction process with an empty set or a full set. The users also have the ability to view a sample of their current document before saving and exporting it. At the section level, the users have the ability to specify which section is currently handled. They also have the ability to quickly redact or include an entire section. Finally, the users have interface options at an entry level which allow items to be controlled at the maximum possible granularity when needed.

The image seen in Figure 9 provides the layout for interface. At the global level, seen in Figure 9(a), this includes the ability to load multiple ".mht" files as well as remove files that were not intended to be included. The top drop down option displays a list of the files currently available to include. The "Load" button beneath this list loads the currently selected item. If that item is already loaded nothing happens. The lower list displays the files that have already been loaded. From here the users can insure that the required sub-trees are included. If an extra tree has been included it may be removed by selecting the file in the lower list and using the "Remove" button. The "Sample" button generates a sample document based on the currently disclosed items. This is done by passing the list of elements currently selected to disclose to the document constructor. Those items are then placed into the skeleton document and the visual portions are constructed as well. This sample document may be seen through the "Sample" tab. The "Redact" and "Keep All" buttons allow the user to either include or remove all items.

The "Sections" tab, Figure 9(b), allows the users to select from a list the section which they wish to interact with. This is a scrollable list with an entry for each section of the document. Once an item is selected it changes the elements viewable from within the "Entries" tab. The section as a whole may also be redacted or added from that tab.

On the "Entries" tab (Figure 9(c)), the users may swipe items left and right to view each item. At each selection they have the ability to redact or keep the current selection with the Redact/Keep button in the bottom center. The text on this button changes to reflect the current action available. for the selected element. This also serves as a visual cue to the users to determine the current state of the item. The entire section may be redacted or kept with the "Redact/Disclose All" buttons at the bottom. This allows the users to handle sections very quickly by removing sections which will only include a small number of items and adding those items back in. Or by removing entire sections which will not contribute to the current transaction. In a similar fashion to the information extraction on document re-population, each element here is summarized so the users can determine the information presented and decide if they wish to include that information in the current transaction.

The "Sample" tab provides the generated sample from the "Docs" tab which allows the user to visually check over the information to see what would be sent out before finalizing the transaction. A small sample document may be found in Figure 9(d).

CHAPTER V

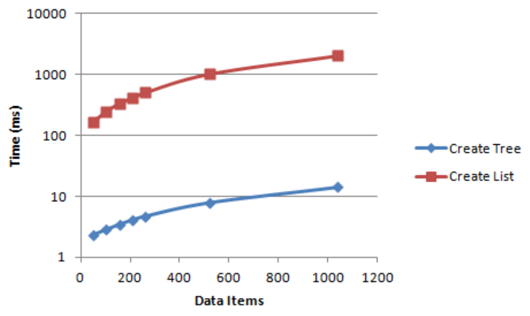
RESULTS

Below, the data is given from various aspects of the system. Results and analysis of document and tree interactions will be provided from the implemented software. This analysis will not include the android application. The application was only implemented and tested on an emulator device as a proof of concept for a redaction interface to the medical elements. There are three main portions to this analysis. The first two focus on the characteristics of the MHT. The speed and memory overhead of interactions with the tree are specifically examined with respect to tree creation, redaction and verification. The first section will serve as a comparison between two methods that provide the desired features of verification and redaction. The second more closely examines the results of the MHT approach. The last section focuses on interaction with the CCD.

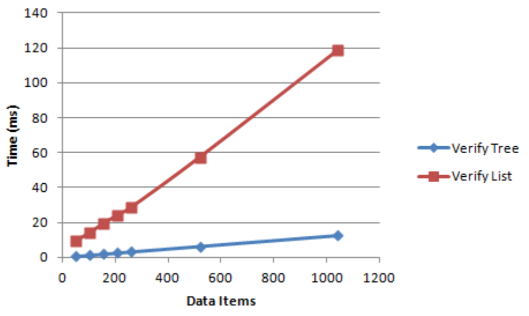
5.1 Approach Comparison

The objective of this research was to find a method for providing verification and redactability in medical records. A brute-force approach for achieving this would be to sign every item that may be released on its own. Each element would be completely independent of all other items and could be verified when required. This first section will examine the differences between these approaches.

For these results, the same set of elements were used from a sample CCD. The initial count of items produced was 52. To increase the count the elements were duplicated for different incremental values up to 1040. It is acceptable to duplicate the items because the concern here is not what elements are handled but how long it takes to handle elements.



(a) Creation



(b) Verification

Figure 10: MHT vs. List of Signatures

5.1.1 Structure Creation

To create this new list structure all items must be hashed and signed. The signed hashes and the items are stored together for transportation and then inserted at the end of a list. The Merkle hash tree approach, as discussed earlier, involves calculating the entire tree of hashes. This number can be approximated to twice as many hashes as elements in a binary tree. And, finally, the root hash is signed once.

The time comparison of these approaches may be found in Figure 10(a). Note that the time is shown in log scale.

From the figure the approach of signing all items and creating the tree may be seen to be about one hundred times as expensive. The reason for this difference comes from the fact that signing an element is a very computationally expensive operation while a hash is relatively cheap in comparison. In larger sets the cost surpasses an entire second to sign these items. When considering real world scenarios these signatures would be done for large sets of medical data. Large numbers of people would provide hundreds or thousands of medical items that need to be signed. The time saved by using a Merkle hash tree would very quickly add up.

This test was performed on a single layer hash tree. If there were to be two signatures on the data, this would need to be replicated by perhaps signing the hash value twice in the list. This would almost double the time to create the list of signatures while adding only a few hash operations and another single signature to the MHT approach.

5.1.2 Structure Verification

The verification of the list is similar to the creation. All items must be hashed again to compare the expected hash value with the provided, signed value. The Merkle hash tree must compare all hashes provided in the tree with the expected ones as well as verify the root hash. These results may be found in Figure 10(b).

The time difference is not as large here as verification of the tree is almost the same cost as creating it. All of the hash values must be created again but signature verification is less expensive computationally than signature creation. This difference in computation cost will be further analyzed during the in depth look at the results for the MHT.

5.1.3 Redaction

Redaction in the list involves checking every item in the list for a match to the item to be redacted. The MHT approach performs a depth first search across all leaves until every leaf is searched or the item is found and redacted. To get a worst case redaction scenario a non-existing element was requested for redaction. This causes the list to search to the end as well as the tree. The time comparisons may be found in Figure 11. This test was performed to demonstrate that the cost of searching the tree during redactions does not outweigh the initial cost benefit of constructing the tree.

The two approaches are found to be similar in cost. Surprisingly the redaction in the tree performs slightly better than redaction in the list. This may be for a couple of reasons. While there are twice as many calls within the tree redaction, a call to redact from every single node, only the nodes that have data items in them check to see if the strings provided a match. Also the list is searched in a loop while the tree

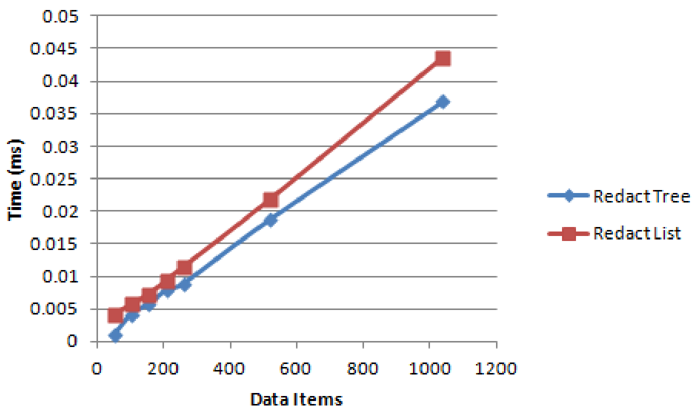


Figure 11: Redaction Comparison

is searched in a single high level call with recursion searching the entire tree. In both cases the redaction times for a single search are only fractions of a millisecond.

5.2 *Detailed MHT Results*

Now that the advantage to this approach has been established, a further examination into the cost of this approach will be given. To put these next few sections into perspective a reiteration of the personal health record scenario is provided. In this scenario, various health care providers produce relevant records for a particular patient. The patient may keep all of these records and compile them into a tree upon the need for a medical transaction. Once this tree is compiled the patient should redact any nonessential elements from the structure to maintain minimum required disclosure. This structure would then need to be verified by the third party.

5.2.1 Tree Creation

As mentioned in the previous sections the advantage to this approach is the use of hash operations over signatures to provide verification and redaction. To demonstrate the cost differences the creation of the structure was broken down into two portions. The first is the time required to fully construct the tree and calculate the hashes. The second is the time required to sign the root. As seen from Figure 12 the time required

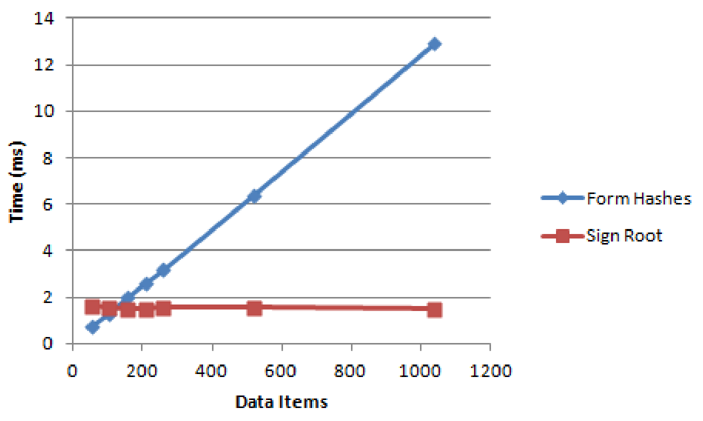


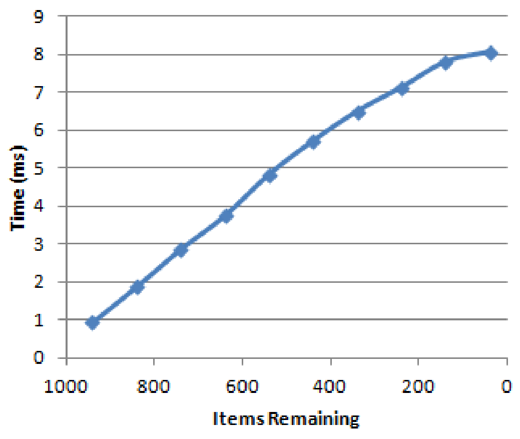
Figure 12: Tree Construction

for a single signature may be approximated by the time required to construct a tree with 200 leaf nodes. To sign 200 items it would take about 400 milliseconds at 2 ms per signature. This may be verified through re-examination of Figure 10(a).

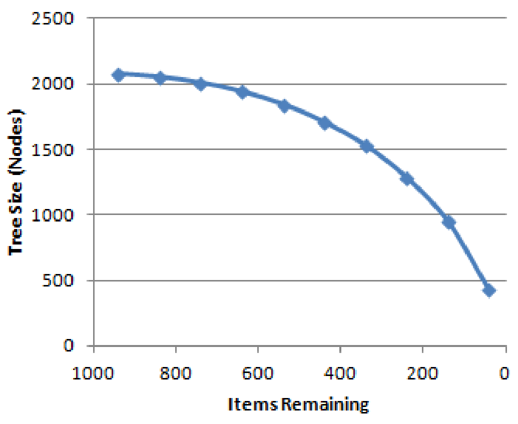
However, to form this structure with up to 1000 items it may be seen from the figure to only require about 15 ms, when including a signature. This is a minimal overhead for even the extreme case demonstrated. The fraction of a second required to construct the tree can be seen negligible when considering the fact that this structure may be used multiple times for every construction process. In most cases, the construction times would be seen from the lower end of this figure. While it is possible for a healthcare provider to produce a very large set of data for a patient's health, this would be the exception rather than the rule. And to the patient, constructing a global tree from 10 different sources with 100 items in each sub-tree may be viewed as constructing a tree with 10 leaves as the patient would not need to calculate hashes within the sub-trees.

5.2.2 Data Redaction

The next step in the logical flow of the data transaction brings up the concept of data redaction overhead. The results in Figure 13 demonstrate the time and memory overhead for different levels of redaction. These results were obtained by equally



(a) Time



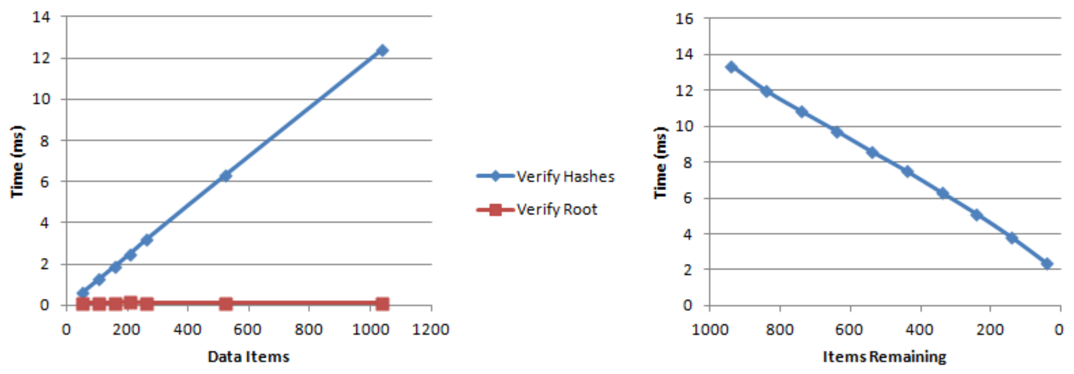
(b) Nodes

Figure 13: Redaction

redacting elements from 20 sub-trees placed in a larger global tree. The initial element count was 1040 for all results.

As seen in Figure 13(a) the time required to redact even 1000 items from the tree was about 8 ms. If doing a manual redaction, the time to make the decisions about which items to redact would be much larger than this. Also if using a policy file to determine which items may be redacted, such a decision would likely require at least an equal amount of time.

The visual found in Figure 13(b) provides an estimate for the number of nodes remaining after a redaction. This is relevant because the remaining nodes provide a look into the amount of data required to be sent on a given transaction. The reason for the change in number of nodes comes from the fact that, as items are redacted, certain hash values and nodes are no longer required for the verification. As the figure demonstrates the number of nodes begins to drop quickly after approximately half of the tree is gone. It should be noted that the most efficient storage may always be found at 0 redactions. With a full tree the efficiency rate is about 2 nodes per leaf item. While at near full redaction the ratio is close to 40 items at 400 nodes remaining. This is due to the number of hashes required at each level to maintain a verifiable tree. The extension of altering the number of children nodes could possibly



(a) Single Root (b) Multi-Root

Figure 14: Verification

have a positive impact on this efficiency rate.

5.2.3 Tree Verification

The last step for the medical transaction would be the verification process. As mentioned earlier this process is similar in fashion to that of creation. This is because the hashes must be re-calculated and compared and finally the root signatures must be verified. This is demonstrated in Figure 14(a) in comparison to Figure 12. The primary difference in the two graphs is the signature time and the verification time. This also accounts for the difference mentioned earlier in the signature list creation and verification. The total verification time even in the worst case is found to be less than 14 ms which should be an acceptable cost for the benefits added by this structure.

To demonstrate the verification cost in a multiple level tree the Figure 14(b) is provided. This verification time was obtained as part of the redaction tests. Initially a tree with 1040 items was given and redactions were performed in increments of 100. The verification time on these different size trees is provided. The increase in time may be seen due to the overhead left from some of the redacted nodes. Even though the elements remaining changes by 100 the size of the tree to be verified does not fully reflect this decrease. This was demonstrated in the Figure 13(b). Also the number of root nodes to verify the signatures in was increased to 21 instead of 1. These characteristics account for the slight time increase; however, the total computational time was found to be about 13 ms in the worst case of 940 elements.

5.3 CCD Results

To demonstrate the system in an expected scenario sample CCDs were loaded to extract the data and store the created trees for "later use". A list of 206 CCDs were used in the following experiment.

5.3.1 Data Extraction

To demonstrate an automated use of converting the XML CCD to a MHT the list of documents was loaded. Each document was then processed for relevant health information to extract. This information was extracted and stored randomly within a corresponding MHT. The hashes were then formed and the root signed with an arbitrary private key. This tree was then saved for later use.

The total average time per document was 312 ms with the average number of elements per document at about 190. These documents had been de-identified so the average number of elements could be seen to be slightly more than this on a real data set. It should be noted that the majority of this time was spent in data extraction. The average time spent in data extraction was 260 ms which comes to about 83% of the time.

The document with the maximum number of data items from the set, which was 828, provided the largest total run time and extraction time. The percentage time spent on extraction was about 85% which is comparable to the average. This result highlights the benefit to creating the tree and the document in parallel to avoid the data extraction. The average creation time of this set would have dropped to 52 ms without the need for data extraction which is a very low overhead cost for creating such a useful structure.

CHAPTER VI

CONCLUSION

The work presented in this thesis details theory and implementation of a signature scheme that can help usher the medical industries records into the electronic domain. Through the use of Merkle Hash Trees to sign the data stored within Continuity of Care Documents these records maintain the verifiable nature of a standard electronic signature with the added feature of redaction. And this may all be achieved at a low overhead cost by using hash functions which are computationally cheap when compared to electronic signatures.

This development allows for the large quantities of health information to be more easily distributed in a verifiable manner while keeping a minimum disclosure policy required by many common scenarios. The scenarios examined here provide compelling arguments for such an implementation. This method would go a long way toward a true implementation for patient controlled disclosure, medical research use, or data provenance needs.

APPENDIX A

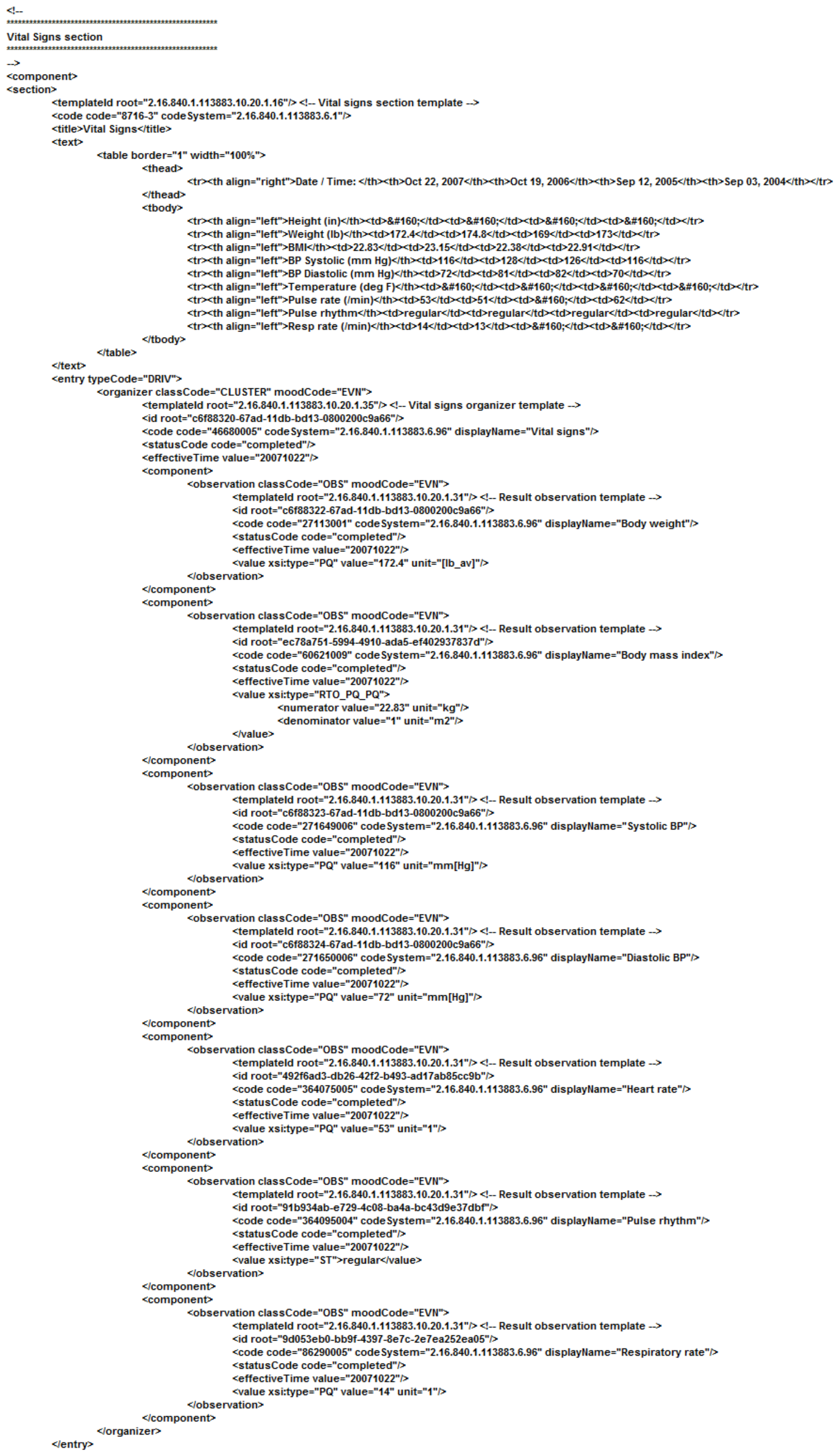
XML VIEW

The information seen in Figure 15 provides a sample view of the XML form of the CCD. The visual table is represented at the top with a single entry provided beneath it. This entry represents all of the vitals for a single date. The entry contains multiple components which each represent a result from the vitals test on that date.

At the very top of this section the templateId may be seen. This information is used as the tag for all elements originating from the vitals section. Each section may have multiple templateId's if the CCD is conforming to multiple sets of standards but the 2.16.840.1.113883.10.20.1 portion references the CCD specifically with the last element describing the type of CCD portion represented. The entry and each component within the entry also have templateId's. These values may be used across different sections. For example the results data and the vitals data may be represented in a similar fashion and use the same entry template or component template.

The visual table may be seen in the "text" node. In this example it was represented as a table. But there is no requirement for this and the data may be stored differently or in a different order. Also the display names provided may be different across different implementations or the measurements provided. Even within this example the patient's weight was labeled as "Weight" in the table but for the weight component within the entry the display name is provided as "Body Weight". With that said all information present within the first column of the table can be found represented within the entry provided. An entry of #160 within the table represents an empty field.

This sample demonstrates a possible need for the extension of data dependencies for the vitals section in particular. Each of these results could be stored in their own entry and handled like all other entries but when presented in this manner the outer portion of the entry needs to be present with any of the inner portions that are disclosed. With the data dependencies implemented for this entry a release policy could be formed such that the outer portion must be released to release any of the inner portions of the entry.



```xml
<!--
********************************************************
Vital Signs section
********************************************************
-->
<component>
<section>
    <templateId root="2.16.840.1.113883.10.20.1.16"/> <!-- Vital signs section template -->
    <code code="8716-3" codeSystem="2.16.840.1.113883.6.1"/>
    <title>Vital Signs</title>
    <text>
        <table border="1" width="100%">
            <thead>
                <tr><th align="right">Date / Time: </th><th>Oct 22, 2007</th><th>Oct 19, 2006</th><th>Sep 12, 2005</th><th>Sep 03, 2004</th></tr>
            </thead>
            <tbody>
                <tr><th align="left">Height (in)</th><td>&#160;</td><td>&#160;</td><td>&#160;</td><td>&#160;</td></tr>
                <tr><th align="left">Weight (lb)</th><td>172.4</td><td>174.8</td><td>169</td><td>173</td></tr>
                <tr><th align="left">BMI</th><td>22.83</td><td>23.15</td><td>22.38</td><td>22.91</td></tr>
                <tr><th align="left">BP Systolic (mm Hg)</th><td>116</td><td>128</td><td>126</td><td>116</td></tr>
                <tr><th align="left">BP Diastolic (mm Hg)</th><td>72</td><td>81</td><td>82</td><td>70</td></tr>
                <tr><th align="left">Temperature (deg F)</th><td>&#160;</td><td>&#160;</td><td>&#160;</td><td>&#160;</td></tr>
                <tr><th align="left">Pulse rate (/min)</th><td>53</td><td>51</td><td>&#160;</td><td>62</td></tr>
                <tr><th align="left">Pulse rhythm</th><td>regular</td><td>regular</td><td>regular</td><td>regular</td></tr>
                <tr><th align="left">Resp rate (/min)</th><td>14</td><td>13</td><td>&#160;</td><td>&#160;</td></tr>
            </tbody>
        </table>
    </text>
    <entry typeCode="DRIV">
        <organizer classCode="CLUSTER" moodCode="EVN">
            <templateId root="2.16.840.1.113883.10.20.1.35"/> <!-- Vital signs organizer template -->
            <id root="c6f88320-67ad-11db-bd13-0800200c9a66"/>
            <code code="46680005" codeSystem="2.16.840.1.113883.6.96" displayName="Vital signs"/>
            <statusCode code="completed"/>
            <effectiveTime value="20071022"/>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="c6f88322-67ad-11db-bd13-0800200c9a66"/>
                    <code code="27113001" codeSystem="2.16.840.1.113883.6.96" displayName="Body weight"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="PQ" value="172.4" unit="[lb_av]"/>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="ec78a751-5994-4910-ada5-ef402937837d"/>
                    <code code="60621009" codeSystem="2.16.840.1.113883.6.96" displayName="Body mass index"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="RTO_PQ_PQ">
                        <numerator value="22.83" unit="kg"/>
                        <denominator value="1" unit="m2"/>
                    </value>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="c6f88323-67ad-11db-bd13-0800200c9a66"/>
                    <code code="271649006" codeSystem="2.16.840.1.113883.6.96" displayName="Systolic BP"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="PQ" value="116" unit="mm[Hg]"/>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="c6f88324-67ad-11db-bd13-0800200c9a66"/>
                    <code code="271650006" codeSystem="2.16.840.1.113883.6.96" displayName="Diastolic BP"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="PQ" value="72" unit="mm[Hg]"/>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="492f6ad3-db26-42f2-b493-ad17ab85cc9b"/>
                    <code code="364075005" codeSystem="2.16.840.1.113883.6.96" displayName="Heart rate"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="PQ" value="53" unit="1"/>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="91b934ab-e729-4c08-ba4a-bc43d9e37dbf"/>
                    <code code="364095004" codeSystem="2.16.840.1.113883.6.96" displayName="Pulse rhythm"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="ST">regular</value>
                </observation>
            </component>
            <component>
                <observation classCode="OBS" moodCode="EVN">
                    <templateId root="2.16.840.1.113883.10.20.1.31"/> <!-- Result observation template -->
                    <id root="9d053eb0-bb9f-4397-8e7c-2e7ea252ea05"/>
                    <code code="86290005" codeSystem="2.16.840.1.113883.6.96" displayName="Respiratory rate"/>
                    <statusCode code="completed"/>
                    <effectiveTime value="20071022"/>
                    <value xsi:type="PQ" value="14" unit="1"/>
                </observation>
            </component>
        </organizer>
    </entry>
```

Figure 15: XML Vitals

REFERENCES

[1] "Health Level 7 International." http://www.hl7.org/implement/standards/cda.cfm.

[2] "Microsoft Health Vault." www.microsoft.com/en-us/healthvault/.

[3] "My HealtheVet." https://www.myhealth.va.gov/index.html.

[4] "NHIN Direct." wiki.directproject.org/.

[5] Ahmed, M. and Ahamad, M., "Protecting health information on mobile devices," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, pp. 229–240, ACM, 2012.

[6] Ateniese, G., Chou, D., De Medeiros, B., and Tsudik, G., "Sanitizable signatures," *Computer Security–ESORICS 2005*, pp. 159–177, 2005.

[7] Bauer, D., Blough, D., and Cash, D., "Minimal information disclosure with efficiently verifiable credentials," in *Proceedings of the 4th ACM workshop on Digital identity management*, pp. 15–24, ACM, 2008.

[8] Bauer, D., Blough, D., and Mohan, A., "Redactable signatures on data with dependencies and their application to personal health records," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pp. 91–100, ACM, 2009.

[9] Brzuska, C., Busch, H., Dagdelen, O., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., and others, "Redactable signatures for tree-structured data: Definitions and constructions," in *Applied Cryptography and Network Security*, pp. 87–104, Springer, 2010.

[10] Bull, L., McG. Squire, D., and Zheng, Y., "A hierarchical extraction policy for content extraction signatures," *International Journal on Digital Libraries*, vol. 4, no. 3, pp. 208–222, 2004.

[11] Bull, L., Stanski, P., and Squire, D., "Content extraction signatures using XML digital signatures and custom transforms on-demand," in *Proceedings of the 12th international conference on World Wide Web*, pp. 170–177, ACM, 2003.

[12] Cadenhead, T., Khadilkar, V., Kantarcioglu, M., and Thuraisingham, B., "Transforming provenance using redaction," in *Proceedings of the 16th ACM symposium on Access control models and technologies*, pp. 93–102, ACM, 2011.

[13] Clarke, D., Breen, L., Jacobs, M., Franklin, R., Tobota, Z., Maruszewski, B., and Jacobs, J., “Verification of data in congenital cardiac surgery,” *Cardiology in the Young*, vol. 18, no. S2, pp. 177–187, 2008.

[14] Haas, S., Wohlgemuth, S., Echizen, I., Sonehara, N., and Müller, G., “Aspects of privacy for electronic health records,” *International Journal of Medical Informatics*, vol. 80, no. 2, pp. e26–e31, 2011.

[15] Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., and Yao, D., “Efficient signature schemes supporting redaction, pseudonymization, and data deidentification,” in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pp. 353–362, ACM, 2008.

[16] Izu, T., Kanaya, N., Takenaka, M., and Yoshioka, T., “PIATS: A partially sanitizable signature scheme,” *Information and Communications Security*, pp. 72–83, 2005.

[17] Izu, T., Kunihiro, N., Ohta, K., Takenaka, M., and Yoshioka, T., “A sanitizable signature scheme with aggregation,” *Information Security Practice and Experience*, pp. 51–64, 2007.

[18] Johnson, R., Molnar, D., Song, D., and Wagner, D., “Digital signatures-homomorphic signature schemes,” *Lecture Notes in Computer Science*, vol. 2271, pp. 244–262, 2002.

[19] Kaelber, D. and Pan, E., “The value of personal health record (PHR) systems,” in *AMIA Annual Symposium Proceedings*, vol. 2008, p. 343, American Medical Informatics Association, 2008.

[20] Kotz, D., Avancha, S., and Baxi, A., “A privacy framework for mobile health and home-care systems,” in *Proceedings of the first ACM workshop on Security and privacy in medical and home-care systems*, pp. 1–12, ACM, 2009.

[21] Lim, S. and Lee, H., “A short and efficient redactable signature based on RSA,” *ETRI Journal*, vol. 33, no. 4, 2011.

[22] Merkle, R., “Protocols for public key cryptosystems,” in *IEEE Symposium on Security and Privacy*, vol. 122, 1980.

[23] Merkle, R., “A certified digital signature,” in *Advances in Cryptology-CRYPTO’89 Proceedings*, pp. 218–238, Springer, 1990.

[24] Miyazaki, K., Hanaoka, G., and Imai, H., “Digitally signed document sanitizing scheme based on bilinear maps,” in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 343–354, ACM, 2006.

[25] Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., and Imai, H., “Digitally signed document sanitizing scheme with disclosure condition control,” *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, pp. 239–246, 2005.

[26] Nazi, K., Hogan, T., Wagner, T., McInnes, D., Smith, B., Haggstrom, D., Chumbler, N., Gifford, A., Charters, K., Saleem, J., and others, “Embracing a health services research perspective on personal health records: lessons learned from the VA My HealtheVet system,” *Journal of general internal medicine*, vol. 25, pp. 62–67, 2010.

[27] Pöhls, H., Samelin, K., and Posegga, J., “Sanitizable signatures in XML signatureperformance, mixing properties, and revisiting the property of transparency,” in *Applied Cryptography and Network Security*, pp. 166–182, Springer, 2011.

[28] Polivy, D. and Tamassia, R., “Authenticating distributed data using web services and XML signatures,” in *Proceedings of the 2002 ACM workshop on XML security*, pp. 80–89, ACM, 2002.

[29] Ramakrishnan, N., Hanauer, D., and Keller, B., “Mining electronic health records,” *Computer*, vol. 43, no. 10, pp. 77–81, 2010.

[30] Rau, H., Hsu, C., Lee, Y., Chen, W., and Jian, W., “Developing electronic health records in Taiwan,” *IT professional*, vol. 12, no. 2, pp. 17–25, 2010.

[31] Samelin, K., Pöhls, H., Bilzhause, A., Posegga, J., and de Meer, H., “Redactable signatures for independent removal of structure and content,” *Information Security Practice and Experience*, pp. 17–33, 2012.

[32] Schnipper, J., Gandhi, T., Wald, J., Grant, R., Poon, E., Volk, L., Businger, A., Siteman, E., Buckel, L., and Middleton, B., “Design and implementation of a web-based patient portal linked to an electronic health record designed to improve medication safety: the patient gateway medications module,” *Informatics in primary care*, vol. 16, no. 2, pp. 147–155, 2008.

[33] Slamanig, D. and Rass, S., “Generalizations and extensions of redactable signatures with applications to electronic healthcare,” in *Communications and Multimedia Security*, pp. 201–213, Springer, 2010.

[34] Slamanig, D. and Stingl, C., “Disclosing verifiable partial information of signed CDA documents using generalized redactable signatures,” in *e-Health Networking, Applications and Services, 2009. Healthcom 2009. 11th International Conference on*, pp. 146–152, IEEE, 2009.

[35] Steinfeld, R., Bull, L., and Zheng, Y., “Content extraction signatures,” *Information Security and CryptologyICISC 2001*, pp. 163–205, 2002.

[36] Suenaga, T. and Takada, A., "Layered secure medical information exchange platform," in *Information Technology Applications in Biomedicine, 2007. ITAB 2007. 6th International Special Topic Conference on*, pp. 157–160, IEEE, 2007.

[37] Weppner, W., Ralston, J., Koepsell, T., Grothaus, L., Reid, R., Jordan, L., and Larson, E., "Use of a shared medical record with secure messaging by older patients with diabetes," *Diabetes Care*, vol. 33, no. 11, pp. 2314–2319, 2010.

[38] Wu, Z., Hsueh, C., Tsai, C., Lai, F., Lee, H., and Chung, Y., "Redactable signatures for signed CDA documents," *Journal of Medical Systems*, pp. 1–14, 2010.