Pseudonymization with Metadata Encryption for Privacy-Preserving Searchable Documents

Johannes Heurix
SBA Research
jheurix@sba-research.org

Michael Karlinger
JKU Linz
karlinger@dke.jku.at

Thomas Neubauer
Vienna University of Technology
neubauer@ifs.tuwien.ac.at

Abstract

The average costs of data leakage are steadily on the rise. As a consequence, several data security and access control mechanisms have been introduced, ranging from data encryption to intrusion detection or role-based access control, doing a great work in protecting sensitive information. However, the majority of these concepts are centrally controlled by administrators, who are one of the major threats to corporate security. This work presents a security protocol for data privacy that is strictly controlled by the data owner. Therefore, we integrate pseudonymization and encryption techniques to create a methodology that uses pseudonyms as access control mechanism, protects secret cryptographic keys by a layer-based security model, and provides privacy-preserving querying.

1. Introduction

In recent times where the quantities of stored data are steadily on a rise, keeping these vast amounts of information secure has become a major challenge. Sensitive corporate data must be protected at all costs from being leaked to unauthorized persons; otherwise organizations have to face massive direct and indirect costs. For example, the leaking of construction plans for a critical component to a competitor may completely set back the company that spent considerable amounts of money on the development; a minor financial institute may not be able to deal with the loss of customer confidence resulting from a security incident involving the theft of the customers' account information. Not only companies but also individuals have to be concerned with data security: An individual's health information leaked to the wrong person may result in severe adverse consequences for this person. Sensitive information about your health status may cause insurance companies to deny coverage. In the e-health sector, health information is often shared with multiple parties, resulting in a significant compromise in patients' privacy [1]. Although security is considered as a critical factor with health information systems, security gaps still exist [2].

In the past, several effective data security and access control mechanisms have been introduced, ranging from data encryption to intrusion detection or role-based access control. However, what the majority of these concepts are centrally controlled by administrators, who define which persons are allowed to access which data. As long as these mechanisms are not circumvented while the administrators are trusted, one can expect an adequate level of data security and privacy. But careless configuration or implementation may result in holes in the security architecture. Especially internal attackers, e.g., disgruntled employees, are a major threat to corporate security when exceeding their access rights and leaking sensitive information to the highest bidder. The most dangerous of the internal adversaries are malicious administrators with their extended privileges, usually endowed with full access rights to fulfill their jobs. The majority of current security concepts cannot protect against this type of attacker. But also external attackers exploiting weaknesses in corporate security layers are able to acquire direct access to sensitive data, unless explicitly protected, as incidents involving Citigroup and Heartland Payment Systems and, more recently, the infamous SONY hack have demonstrated so impressively [3].

Therefore, this work presents a security architecture for data privacy that is strictly controlled by the data owner, i.e., the data owner decides who is granted access to the data, which takes away the required trust in the administrators, especially database administrators. As relying on a single security strategy has its downsides, we integrate pseudonymization and encryption techniques to overcome their individual shortcomings and create a protocol that uses pseudonyms as access control mechanism, protects secret cryptographic keys by a layer-based security model, and supports privacy-preserving querying.

978-0-7695-4525-7/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2012.491

2. Background

From a conceptual point of view, two approaches on how to deal with data confidentiality and data leakage prevention exist: (i) limit access by a dedicated access control system and (ii) modify and persist the data records themselves such that a potential attacker does not gain any useful information, i.e., data masking. Resource modification in this manner can be achieved by either making the data unreadable for unauthorized parties (encryption) or by disassociating individual data items. Data disassociation assumes that the main property of data records that needs to be kept secret is the association between the items, not the items themselves (e.g., if the individual items are publicly available). Therefore, data disassociation "encrypts and delinks the data held about an individual from the individual's identity" [4]. Anonymization and the similar pseudonymization are examples of techniques based on data disassociation.

Traditional Access Control (in the context of this work) refers to limiting access to resources by a dedicated access control module, defining and deciding which actors are allowed to access which resources, or in other words, explicit access control. Role-based access control (RBAC) for example decides on the role the actor currently impersonates. The access rights are defined as rules or policies and can be expressed, e.g., in the eXtensible Access Control Markup Language (XACML). For RBAC, a policy expressed in XACML [5] includes (among others) elements for the particular role (actor), the resources (object), and the permitted actions on the resources (create, retrieve, etc.). These policies are created by the policy administration point (PAP). Access requests are checked against one or more policies and thus granted or denied by the policy decision point (PDP) and enforced by the policy enforcement point (PEP).

Disassociation techniques include anonymization which can be achieved by depersonalization, i.e., the systematic removal of the individual's identifying information from data records such that the records cannot be traced back to the corresponding individual. *K-anonymity* [6] deals with the existence of quasi-identifiers (i.e., elements that are not identifying per se, but may be when grouped together, such as ZIP code combined with last name and birth date) by using generalization and suppression techniques to create equivalence groups. Some extensions of the basic k-anonymity approach deal with the problem of similarity of data tuples within an equivalence group (*l-diversity* [7]) and the distribution and semantic distance of specific sensitive attributes within an equivalence block in the complete dataset (*t-closeness* [8]) to further reduce the risk of re-identification. While anonymization is non-reversible, pseudonymization is a similar technique with the difference that identifying information is not permanently deleted but separated from the data records and referenced by a specifier, the pseudonym. Thereby, the process of depersonalization is reversible under specified and controlled circumstances, i.e., when knowing a particular secret. Pseudonymization is often used in identity management (e.g., [9]) but is also applied in other application areas as well, such as e-health. In this context, pseudonyms are generally used as 'secret' links between patients and health records where the links are only recoverable when being authorized (cf. [10], [11], [12]).

Encryption is the straightforward approach to shield sensitive data from unauthorized glances. The main issue here is how to efficiently query within encrypted data. The naive solution is to transmit the entire encrypted database to a trusted machine where the data is decrypted and then processed as usual. A more efficient and sophisticated approach involves special encryption techniques or some kind of pre-calculated index created by the data owner or data provider and post-filtering the result set. The simplest form is a hash-based or encryption-based index over individual attribute values. In [13], [14], encrypted table rows are stored along with a set of hash-based indexes, depending on which of the table columns are required for queries. Another approach is described in [15] where buckets are introduced, spanning over a pre-defined range of the attributes' domain values. Each bucket is assigned an identifier which serves as index. Other approaches exploit the hierarchical structure of XML documents. For example, in [16], [17], [18], an XML document is stored as a set of (disjoint) document fragments, and crypto-indexes are used to facilitate the search. To answer a query on the structure or the content of an XML document, the crypto-indexes are scanned and all matching document fragments are transmitted to the client. The client then decrypts the fragments and performs some post-processing on the retrieved fragments in order to obtain the final query result.

Limitations. Traditional access control mechanisms are secure as long as the architecture is intact. If the access control module is circumvented by an external attacker, all data records are prone to be leaked due to the lack of further protection mechanisms. Actors like system or database administrators usually have unrestricted access to all sensitive information as well. Traditional access control methods are the predominant protection mechanisms implemented in electronic health care where RBAC is usually implemented, such as in Austria's ELGA (electronic health record) or the UK

National Health Service. For instance, the IHE (Integrating the Health Enterprise) standard, which defines how to exchange health data, requires that data is protected adequately by policy-based access control, but encryption or disassociation is not mentioned [19]. The HIPAA (Health Insurance Portability and Accountability Act) does not dictate mandatory encryption either; it is stated as optional [20]. In health care, most RBAC systems have exception mechanisms to circumvent normal access control which are often overused [21]. Privacy violation incidents have shown that hospital employees do exploit their (technical) access rights [22]. Disassociation and encryption alter the data structure of the stored information, also protecting against internal attackers, as long as the involved crypto keys are secure, but they have other drawbacks: Anonymization techniques are accompanied by loss of information and data accuracy, thus limiting data expressiveness. They cannot be reversed either, restricting their applicability to secondary use of the data pool only (e.g., for statistical or research purposes). Data encryption on the other hand prevents efficient secondary use unless explicitly decrypted, which can be a major disadvantage, especially in e-health where secondary use of medical records for research is an important factor. Data sharing, i.e., access authorization, is also tricky to be handled, requiring either the redundant storage of encrypted data (if re-encrypted with the authorized person's personal key) or sharing the secret decryption key itself. The latter makes de-authorizations rather tedious, demanding re-encryption of the particular record and re-issuing the new decryption key to all other still authorized persons. Decryption can also be a performance-related issue when the records are very large and processing power is limited. Pseudonymization of the sensitive data records supports both privacy-preserving primary and secondary use as long as the records are diligently depersonalized. The issue with pseudonymization is how to realize privacy-preserving querying: The metadata for searching must not contain any sensitive keywords to prevent leakage of critical information, in other words, the domain of keywords must be standardized and highly-structured. No arbitrary and therefore potentially compromising keywords should be allowed.

3. PERiMETER – A Hybrid Pseudonymization and Encryption Approach

To overcome the individual weaknesses of pseudonymization and data encryption, we propose PERiMETER (Pseudonymization and pERsonal METadata EncRyption). It is an extension of previous work [23], [24], [25] and provides the following contributions:

- Fragment the sensitive documents into non-critical fragments and store them in cleartext in a non-hierarchical fashion.
- Pseudonymize the fragments, i.e., assign each fragment a number of pseudonyms to be used as access identifiers and access authorization tickets. For each user who is authorized for a particular fragment, a new pseudonym is created.
- Retain the documents' organizational structure and associations as extracted metadata organized into an XML document, along with queryable document descriptions and arbitrary keywords.
- Instead of encrypting the actual document, only encrypt the metadata with the secret key of the particular data owner to create a personal extended table of contents. This ensures that (initially) the data owner is the only person that can correctly reestablish the links between the document fragments.
- For document access authorizations, forward the corresponding metadata sections to the trusted authorized persons, granting them the ability to identify the associations between the fragments as well. The metadata information then needs to be added to the authorized users' personal metadata databases, encrypted with their corresponding secret keys.
- Use semantic information of the XML metadata to realize an efficient structure- and content-related query process with a schema-aware labeling scheme.

In the following, the basic concepts of our approach are described.

3.1. Pseudonym-Based Access Control

Pseudonyms as document identifiers provide a form of 'traceable anonymity': The usage of pseudonymization relies on the disassociation of particular document fragments. In terms of pseudonymized data, access control does not refer to the traditional definition but refers to the knowledge and the ability to reconnect certain pseudonyms to each other. If someone is authorized to a particular document, this person is able to identify which pseudonyms belong together and, thus, which fragments are part of the specific document. If not authorized, the person cannot identify the correct associations between the fragments. In the following, we refer to the document fragments simply as 'records'.

In PERiMETER, the pseudonyms are randomly selected and not derived from any entity. We distinguish between root and shared pseudonyms: Root pseudonyms are available to the data owner only and represent the main access identifiers. Whenever new records are stored, a new root pseudonym is assigned to each individual record. Shared pseudonyms represent access authorizations for trusted users. The shared pseudonyms are created by the data owner and shared only with the particular trusted user. For each individual access authorization, a new shared pseudonym is created, thus the number of randomly selected and assigned pseudonyms per record is $1 + n_{auth}$ where $0 \leq n_{auth} \leq n_{total}$; n_{total} is the number of all potential authorization grantees and n_{auth} the number of persons currently authorized for the particular record, while usually $n_{auth}^{rec_a} \neq n_{auth}^{rec_b}$, i.e., the number of assigned authorizations need not be the same for all records in the data owner's possession. In Figure 1 the record (in the middle of the lower part) is assigned two pseudonyms via its record identifier. While the data owner has knowledge of both root and shared pseudonyms, the authorized person only has access to the shared pseudonym. Multiple authorizations for the same record but for different persons can be organized under the Authorization section in the owner's personal metadata storage (cf. Section 3.3). In any case, the data owner is the only person that has the knowledge of all authorizations, while the authorized persons are limited to their individual authorization. The data owner also keeps track of all authorized persons via their user identifiers, while the authorized users store the corresponding data owners' identifiers of all authorizations they have received.

3.2. Layer-Based Security Model

PERiMETER makes use of a layer-based security model (see Figure 1) which protects the secret cryptographic key used to encrypt each user's metadata storage: the inner symmetric key (ISK). Each layer is responsible for one step in the data access process. The user has to pass all layers in order to retrieve the actual documents. The outer layer, the authentication layer, is responsible for proving the user's identity. Technically, the outer layer is realized by the outer asymmetric keypair (OPuK, OPK). We propose to use a smart card as security token as explained further below (cf. Section 3.4). The outer private key on the security token is only accessible when entering the correct PIN and is used to decrypt the inner private key (IPK) which in turn is needed for decrypting the inner symmetric key as it is encrypted with the inner public key (IPuK). The (pseudonymization) metadata storage encrypted with the inner symmetric key forms, along with the inner keys (symmetric and asymmetric) and the outer public key, the (middle) authorization layer. The inner keypair is also used for encrypting confidential messages exchanged asynchronously between the users (e.g., notification of a de-authorization). The plaintext pseudonyms attached to the records represent the innermost layer, the concealed data layer. Thus, the protection envelope is basically $\{\{\{own \mapsto psn_1, \dots, psn_n\}_{ISK}\}_{IPuK}\}_{OPuK}$ for the cleartext mappings $psn_1 \longmapsto rec_1, \dots, psn_n \longmapsto rec_n$ where own denotes the data owner, psn_i a particular pseudonym and rec_i the corresponding record it is mapped to.

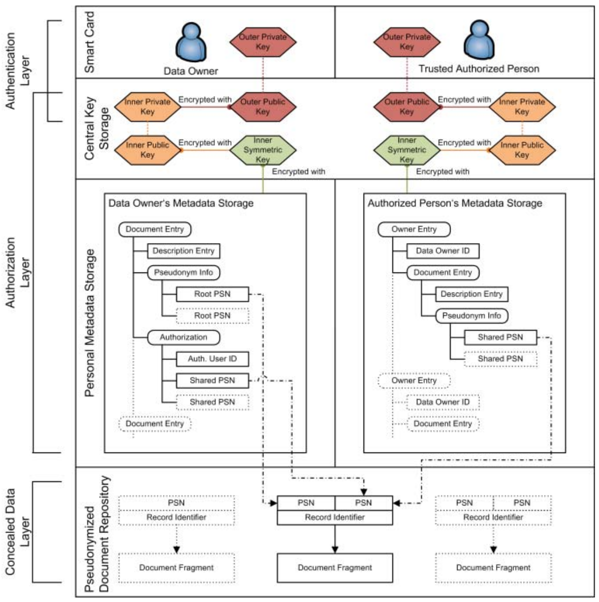


Figure 1: The PERiMETER Approach

3.3. Schema-Aware XML Document Encryption

Each user has her own personal metadata storage acting as personal document directory. The directory is organized as an XML document and is stored encrypted with the particular user's inner symmetric key at the central metadata storage provider [26]. Therefore, it is accessible only by the key-owning user, while the pseudonym/record mappings (see Concealed Data Layer in Figure 1) are accessible by all users. The XML data structure allows organizing the directory entries in a hierarchical manner, as long as the structure corresponds to predefined schema information obtained from an XML schema or DTD. Because each user has her own private store, each user is able to organize the document entries at her discretion, as long as it corresponds to the schema. As query mechanism, we use a schema-aware labeling scheme to represent XML documents [27]. Each node is assigned a unique node label l which encodes the path leading to the node, as well as the node's tag name and type. The content of an XML document is stored at the metadata storage provider in a hash table $H(l) \longmapsto E(v, ISK, n)$ where H is a cryptographic hash function, E is a symmetric encryption algorithm taking the textual value v of the leaf node labeled l as input to be encrypted with key ISK nonced with n. The structure of an XML document is stored in another hash table which essentially represents a B^+-tree over the node labels in the document. To further speed up query performance, secondary index structures can be employed. We employ nonce-based encryption to prevent duplicate ciphers (the same XML content may exist within an XML document at different locations). To insert a new node into a document, first the node label is computed and then new entries are added to the hash tables for storing the structure and the content of the document. Then secondary index structures are updated accordingly.

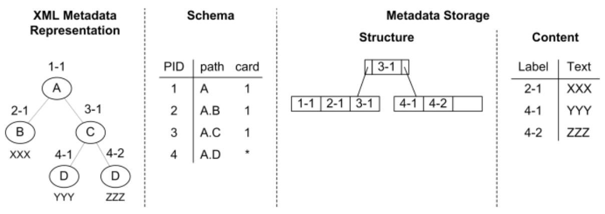


Figure 2: Metadata Storage

In Figure 2, the XML document consists of the text leaf nodes B and D connected by nodes A and C. The schema defines the path and the corresponding identifiers, as well as the cardinality of the particular node. While the document structure is represented as B^+-tree, the content is stored separated from the structure.

Queries are defined in XPath expressions. The supported XPath fragment allows for navigational queries with value-based predicates. Navigational queries are primarily processed by means of querying the structural information about the document, but also highly expressive node labels are exploited to reduce the number of storage access operations. To process value-based predicates, secondary index structures are used if available, or the values of nodes need to be filtered by the retriever. Basically, exact match queries, range queries, or a combination of them can be realized by defining the corresponding XPath expressions, supported by prepared secondary index structures.

3.4. Required Hardware and Overall Architecture

For optimal protection of the secret keys, we propose to use smart cards as user-owned security tokens for authentication (cf. Section 3.2). The combination of smart card and user PIN provides two-factor authentication and is thus significantly more secure than using a simple username/password combination. In this context, the smart card is a secured and tamper-resistant (contact) micro controller card with a secured key storage area and dedicated hardware-based cryptographic engines for common algorithms, such as RSA or AES. As smart cards may be lost or damaged, we employ a backup mechanism based on secret sharing [28]. Apart from the smart cards, the other components of the overall architecture are the pseudonymization and query logic module and the storage provider (cf. Figure 3).

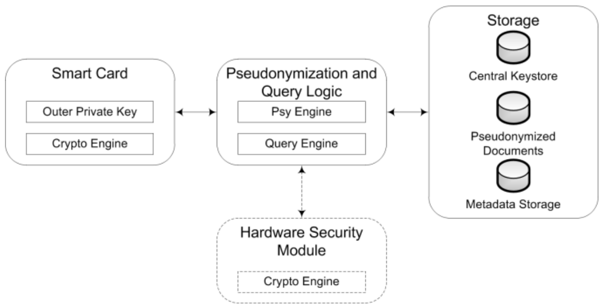


Figure 3: Technical Architecture

The storage provider includes the central keystore, the pseudonymized documents, and the personal metadata storage. The central keystore contains all user keys (apart from the outer private key) where the secret keys are stored encrypted according to the security layer architecture (cf. Section 3.2). The pseudonymized document storage contains the document fragments and associated plaintext pseudonyms, while the metadata storage persists the searchable document metadata organized into encrypted user-specific (virtual) databases. Due to the encrypted status of keystore and metadata content and the pseudonymized data structure of the (payload) documents, the storage provider need not be fully trusted regarding confidentiality.

Concerning the logic module which is responsible for document pseudonymization, querying, and relinking document fragments, we distinguish between two scenarios.

- **Local:** In the local scenario, the logic module is situated at the user's client machine. The smart card may act as primary cryptographic module without relying on the host (i.e., workstation) such that all crypto operations are executed within the secured environment of the card. This also includes that all secret keys (outer and inner private and inner symmetric keys) are available in plaintext only within the smart card.
- **Central:** In the central scenario, the logic module is located at the server (i.e., storage provider), providing the pseudonymization service. In this case, the smart card only acts as authentication token which decrypts the inner private key after being retrieved from the central keystore. The inner private key acts as decryption token and is transferred to the server's side where the inner symmetric key, retrieved from the keystore, is decrypted. The inner symmetric key used for the actual cryptographic operations, thus, always stays at the server's side. In this case, we propose to employ a hardware security module (HSM) to keep the keys protected during use. Similar to smart cards, HSMs are tamper-resistant cryptographic devices with dedicated support for common algorithms, but at a considerably higher performance level.

For mutual authentication, a challenge/response authentication procedure is employed involving the user's outer keypair and the storage provider's asymmetric keypair. In the central scenario, the storage provider's (or server's) public key also protects the user's inner private key during the client/server-transfer. Also in the central scenario with a HSM, security can be further improved by encrypting the pseudonym/record identifier mappings (see Figure 1) with a logic key only known to the HSM. In this case, the following conditions need to be met in order to be able to retrieve a particular document: (i) authenticated user providing her secret inner symmetric key, (ii) the HSM with its logic key, and (iii) access to the metadata storage as well as to the pseudonymized document fragments.

4. Prototype and Case Study

Due to the flexibility of the document metadata/pseudonym combination, numerous potential application domains exist. In general, the pseudonymization technique of PERiMETER excels in any area where relatively large documents exist and full document encryption would result in a considerable performance drawback. However, we identify the e-health sector to be the major application area where the secure storage of medical documents in cleartext has a distinctive advantage: their usage for research purposes in a privacy-preserving manner. While the patients are considered the data owners of their health documents, they are responsible for creating authorizations for trusted health care providers (for primary use) at a need-to-know basis. Provided that during the pseudonymization process the documents were diligently depersonalized, the records can be directly used without any further privacy-preserving measures.

As a proof-of-concept, we implemented a prototype using a combination of Java and .NET technology for the implementation of the main logic module. As smart cards, we use programmable Gemalto .NET V2+ Cards that can be seamlessly integrated with the logic module. As storage provider, we employ a MS SQL Server as key and document storage and a JBOSS application server as metadata storage provider. The prototype's architecture follows the 'local' approach, i.e., the logic's module is executed at the client machine with the smart card as main cryptographic device.

Document fragmentation is handled as follows: the health documents encoded in HL7 CDA (Health Level 7 Clinical Document Architecture) are separated into *identification* and *health* records in that the CDA document's header part corresponds to the identification and the document's body to the health record part (in CDA, document headers include organizational and patient-related information, such as patient's and health care provider's names, identifiers, and others, while the document body section contains the actual health-related information). Each identification record (header section) is assigned root and shared identification pseudonyms, while the health records (body section) are assigned root and shared health pseudonyms as shown in Figure 4. In this data model, the document fragments are always organized in a 1:1 relationship of one identification and one health record, resulting in root and shared pseudonym pairs.

For each (complete) CDA health document, the corresponding metadata entry includes pseudonyms, user identifiers, and record description elements: While the data owner's metadata storage includes all root pseudonym pairs as well as all authorizations represented by shared pseudonym pairs organized under the same node (see Figure 1), the authorized person's view involves only the shared pseudonym pair. Description elements are basically equal in both metadata databases, but can be extended with arbitrary elements by the metadata storage owner if required.

The description elements are document-type-specific, i.e., metadata lab results include different searchable description elements than a medical discharge letter or anamnesis.

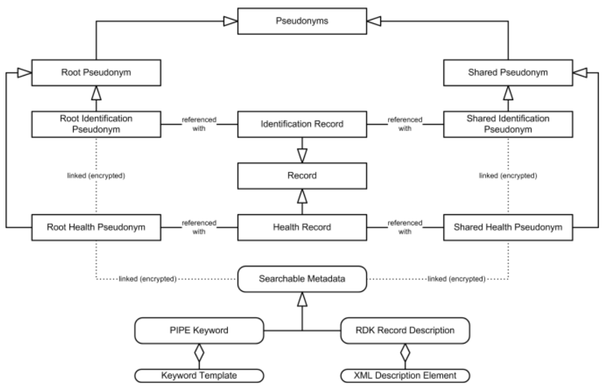
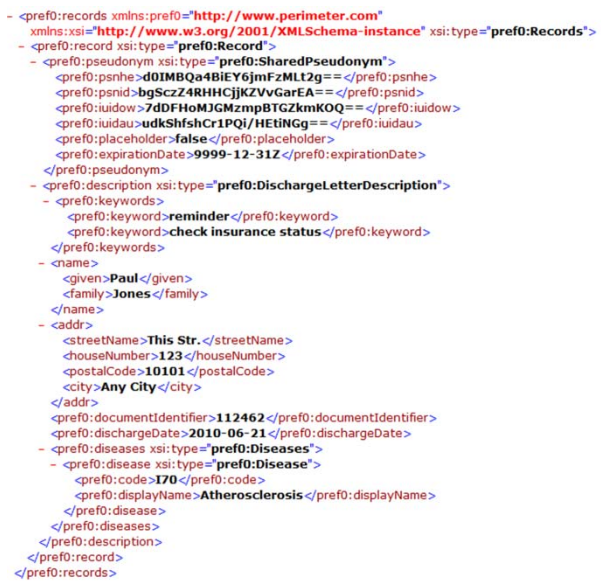


Figure 4: Health Document Data Model

In our exemplary scenario, the fictional patient Paul Jones has been recently discharged from in-patient stay and received a discharge letter. He now authorizes his trusted general practitioner access to the letter. This discharge letter is encoded in HL7 CDA and stored in a pseudonymized state as described above. During the authorization process, a new shared pseudonym pair is created, assigned to the two record identifiers of both fragments, and appended to the corresponding root pseudonym pair in Paul Jones' metadata storage. In addition, the description elements are retrieved and forwarded to the general practitioner, along with the shared pseudonym pair, who adds the new entry to her personal metadata storage and encrypts it with her secret key. Figure 5 illustrates the (plaintext) document description entry from the general practitioner's viewpoint.

Apart from the shared pseudonyms (*psnid* referring to the document header fragment, *psnhe* to the document body) and the (internal) user identifiers (*iuidow* corresponds to the patient's (data owner's) identifier, while *iuidau* refers to the authorized user's identifier, in this case the general practitioner's identifier as authorization grantee), the entry contains general queryable elements, such as the patient's name, as well as elements unique to the *DischargeLetterDescription* type, including discharge date and a list of diagnosed diseases encoded in the ICD10 standard (International Statistical Classification of Diseases and Related Health Problems Rev.10) - I70 for Atherosclerosis. Furthermore, the general practitioner added arbitrary private keywords including *check insurance status*. These arbitrary keywords are only stored at the general practitioner's metadata storage and are thus not visible to anyone except to her.



```
- <pref0:records xmlns:pref0="http://www.perimeter.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="pref0:Records">
  - <pref0:record xsi:type="pref0:Record">
    - <pref0:pseudonym xsi:type="pref0:SharedPseudonym">
        <pref0:psnhe>d0IMBQa4BiEY6jmFzMLt2g==</pref0:psnhe>
        <pref0:psnid>bgSczZ4RHHCjjKZVvGarEA==</pref0:psnid>
        <pref0:iuidow>7dDFHoMJGMzmpBTGZkmKOQ==</pref0:iuidow>
        <pref0:iuidau>udkShfshCr1PQi/HEtiNGg==</pref0:iuidau>
        <pref0:placeholder>false</pref0:placeholder>
        <pref0:expirationDate>9999-12-31Z</pref0:expirationDate>
      </pref0:pseudonym>
    - <pref0:description xsi:type="pref0:DischargeLetterDescription">
      - <pref0:keywords>
          <pref0:keyword>reminder</pref0:keyword>
          <pref0:keyword>check insurance status</pref0:keyword>
        </pref0:keywords>
      - <name>
          <given>Paul</given>
          <family>Jones</family>
        </name>
      - <addr>
          <streetName>This Str.</streetName>
          <houseNumber>123</houseNumber>
          <postalCode>10101</postalCode>
          <city>Any City</city>
        </addr>
        <pref0:documentIdentifier>112462</pref0:documentIdentifier>
        <pref0:dischargeDate>2010-06-21</pref0:dischargeDate>
      - <pref0:diseases xsi:type="pref0:Diseases">
        - <pref0:disease xsi:type="pref0:Disease">
            <pref0:code>I70</pref0:code>
            <pref0:displayName>Atherosclerosis</pref0:displayName>
          </pref0:disease>
        </pref0:diseases>
      </pref0:description>
    </pref0:record>
</pref0:records>
```

Figure 5: Document Description Entity

A typical query expressed in XPath has the elements

```
/child::records
/child::record[child::element(description,
        DischargeLetterDescription)]
/self::*[child::description
        /child::keywords
        /child::keyword = 'check insurance status']
/child::element(pseudonym,SharedPseudonym)
```

and yields the shared pseudonyms of the discharge letter fragments that belong to the patient Paul Jones. With the retrieved pseudonyms, the logic then retrieves the corresponding fragments to restore the original discharge letter document.

Apart from the basic record storage/retrieval and (synchronous) authorization/de-authorization functions, we also implemented extended functionality, such as asynchronous authorizations, which do not require the concurrent physical presence of both the data owner and authorized user (and thus their smart cards) and data adding authorizations, authorizations for new records that are yet to be added in the future. These data adding authorizations involve the creation of placeholders that are replaced with the actual records when available (as in the health care sector, the primary data provider is usually not the patient (as data owner) but a health care provider). Asynchronous operations make use of our notification system involving the inner asymmetric keypair for confidentiality.

5. Analysis

In the following, we analyze PERiMETER's security and performance properties.

5.1. Security

The overall privacy assurance of PERiMETER is based on the following three aspects: crypto key secrecy, query secrecy, and record unlinkability.

Key Secrecy. In PERiMETER, each user has her own set of secret keys where the outer private key always stays at the user's side, while the other secret keys are stored at the central keystore protected by the security layers ($\{ISK\}_{IPuK}, \{IPK\}_{OPuK}$). Considering the tamper-resistance of smart cards and the fact that the outer private key never leaves the card (actually all keys are generated on the card), and assuming PIN secrecy, the outer private key is assumed to be reasonably secure, as are therefore the other secret keys as well. In the local deployment scenario (cf. Section 3.4), in addition to the inner private key, the inner symmetric key is decrypted at the card as well during the login procedure (authentication at the storage provider). If the card is used as main crypto module (i.e., without crypto operations being executed at the host machine such as the user's workstation), all secret keys including the inner symmetric key are in plaintext only within the secure confinements of the card. To achieve the same security property in the central scenario, the inner private key needs to be encrypted with the HSM's public key prior to the transfer to the HSM in order to ensure a seamless 'secured channel' between user-side smart card and server-side HSM.

Query Secrecy. Query secrecy is ensured by the metadata storage and retrieval scheme that stores the document metadata fragmented into individual nodes represented by key/value pairs in a hash table without revealing secret information (see Section 3.3). The keys of the hash table containing the node labels (or node values for reverse lookups/secondary index structures) are calculated by a (salted) hash algorithm for efficient search, while the values of the hash table containing the node values (or node labels for reverse lookups/secondary index structures) are required to be encrypted by a (reversible) symmetric encryption scheme. A $query(H(l))$ containing node label l masked with the hash algorithm H therefore contains no cleartext elements, while the corresponding return value $return(\{v\}_{ISK,n})$ contains the node value encrypted with the inner symmetric key ISK enhanced with the nonce n. Given the cryptographic strength of both the hashing and encryption algorithms and the key secrecy assumption stated above, the stored metadata in general and the queries in specific are arguably secure. The nonce prevents attacks based on frequency analysis on cipher texts.

Record Unlinkability. The main privacy property of PERiMETER, record unlinkability, is provided by the pseudonymized document fragments. The general goal is to mask the relation $user_x \mapsto rec_x$. This is achieved as follows: First the direct mapping is broken up by introducing a set of individual pseudonyms such that $user_x \mapsto psn_x \longmapsto rec_x$. For a second $user_y$ being authorized to access the same record rec_x, the following additional mapping exists: $user_y \mapsto psn_y$ while $(psn_x/psn_y) \mapsto rec_x$. The latter pseudonym/record mappings remain in cleartext whereas the former user/pseudonym relations are persisted in the metadata storages. Therefore, $\{user_x \longmapsto psn_x\}_{ISK_x}$ and $\{user_y \mapsto psn_y\}_{ISK_y}$ where due to the key secrecy assumption $user_x \longmapsto user_y$ (i.e., authorization relation) or $user_x \mapsto psn_y$ cannot be identified, unless $user_x$ is the data owner and $user_y$ the trusted authorized person, in which case $user_x$ has created psn_y as authorization. Therefore, all unauthorized persons cannot identify $user_x \mapsto rec_x$, and, assuming that another $user_z$ has been authorized for rec_x, both authorized persons $user_y$ and $user_z$ have the knowledge of $user_x \mapsto rec_x$, but the authorization relation $user_x \longmapsto user_y$ is hidden for $user_z$ and vice versa.

5.2. Performance

Performance of PERiMETER is largely relying on cryptographic and database operations. While the challenge/response-based authentication procedure is required only once for each login and data retrieval is expected to be executed much more often than data storage, we analyze a typical query execution and document retrieval operation in terms of required encryption/decryption and database retrieval operations.

A document retrieval operation can be broken down into two phases: metadata query execution to identify the correct pseudonyms and the actual records' (i.e., document fragments') retrieval. The latter requires no decryption and depends simply on the number of document fragments to be retrieved. The performance of metadata query execution depends highly on the query's complexity which influences how many nodes within the XML metadata document need to be visited. The counts of hashing, database retrieval, and decryption operations are exactly $count_H = count_R = count_D = count_{visited\ nodes}$. Taking the query and metadata document in Section 4 as an example, first all *record* nodes are visited and checked whether their descendant *description* equals to *DischargeLetterDescription* and then whether one of their *keyword* nodes' values equals to *check insurance status*. If so, both *psnid* and *psnhe* nodes' values are retrieved. Depending on the number of total *record* elements within the XML document, a potentially large number of individual nodes needs to be visited to identify the requested pseudonyms.

To improve the performance of query processing, we implemented two optimization mechanisms: secondary index structures and alternative node fragmentation, the former aimed at reducing the number of visited nodes to *find* the correct entry and the latter to reduce the decryption operations of the actual *values* (i.e., pseudonyms) one is interested in. For the exemplary metadata document, suitable index structures are built on *keyword* and, e.g., *dischargeDate* pointing to *record*. If a query contains both elements (e.g., *keyword* = *reminder* and *dischargeDate* > 2010-01-01), the results are joined to find the *record* matching both elements, thereby significantly reducing the visited nodes. If the query should always return both pseudonyms, *psnid* and *psnhe* can be stored in a single node, further reducing the visited nodes (and thus decryption operations). In addition to these optimizations, a typical caching mechanism is also implemented which retains the recently accessed nodes.

In terms of actual execution time, a complex query requiring tens of seconds (with ISK decryptions directly on the smart card) can be reduced to a couple of seconds with proper indexing and fragmentation, or even to far less than a second (with trusted host and ISK decryption on the workstation in the local scenario, or with an HSM in central scenario).

5.2. Limitations

Successful pseudonymization provided by PERiMETER requires the following preconditions:

- Document fragmentation must be conducted in such a way that the individual fragments do not pose any privacy risk when disclosed. When storing information containing personal data, such as HL7 CDA encoded health records, the body section must not contain any patient-identifying information.
- Effective pseudonymization requires a sufficiently high number of documents to provide unlinkability. Obviously, it makes no sense to pseudonymize a single document into several fragments when the document is the only record stored in the database.

6. Conclusion

Data confidentiality and security is a major issue nowadays as demonstrated by numerous examples of data leakage in recent history. Often internal attackers, such as malicious administrators or disgruntled employees, are responsible for compromising data confidentiality. In this work, we presented PERiMETER, a Pseudonymization and pERsonal METadata EncRyption approach for ensuring data security that specifically addresses internal attackers by minimizing the leakage of sensitive information in case of unauthorized data disclosure, i.e., data theft. It provides (i) key management without the need of external trusted third parties, (ii) secure document storage by pseudonymization, (iii) a pseudonym-based authorization mechanism controlled by the data owners, and (iv) a privacy-preserving query mechanism that efficiently searches within encrypted metadata.

Further work deals with the semantic annotation of specific information such as personal data for automated de-personalization of personal records and automated extraction of useful queryable items from individual document fragments (e.g., dates, document types, domain-specific keywords, etc.).

References

[1] Lechler T., Wetzel S., Jankowski R., Identifying and Evaluating the Threat of Transitive Information Leakage in Healthcare Systems, Proc. of the 44th Hawaii International Conference on System Sciences, 1-10, 2011

[2] Luethi M., Knolmayer G.F., Security in Health Information Systems: An Exploratory Comparison of U.S. and Swiss Hospitals, Proc. of the 42nd Hawaii International Conference on System Sciences, 1-10, 2009

[3] Grocer S., Sony, Citi, Lockheed: Big Data Breaches in History, The Wall Street Journal, June 9, 2011

[4] Eggers, W.D., Government 2.0: Using Technology to Improve Education, Cut Red Tape, Reduce Gridlock, and Enhance Democracy, Rowman and Littlefield, 2007

[5] OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0, Committee Specification 01, 2010

[6] Sweeney L., k-Anonymity: A Model for Protecting Privacy, International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10(5), 557-570, 2002

[7] Machanavajjhala A., Kifer D., Gehrke J., Venkitasubramaniam M., l-Diversity: Privacy Beyond k-Anonymity, ACM Transactions on Knowledge Discovery from Data, 1(1), 2007

[8] Li N., Li T., Vekatasubramanian S., t-Closeness: Privacy Beyond k-Anonymity and l-Diversity, IEEE 23rd International Conference on Data Engineering, 106-115, 2007

[9] Camenisch J., Shelat A., Sommer D., Fischer-Hübner S., Hansen M., Krasemann H., Lacoste G., Lenes R., Tseng J., Privacy and Identity Management for Everyone, Proc. of the 2005 Workshop on Digital Identity Management, 20-27, 2005

[10] Thielscher C., Gottfried M., Umbreit S., Boegner F., Haack J., Schroeders N., Data Processing System for Patient Data, Int. Patent, WO 03/034294 A2, 2005

[11] Noumeir R., Lemay A., Lina J., Pseudonymization of Radiology Data for Research Purposes, Journal of Digital Imaging, 20(3), 284-295, 2007

[12] NEMA, Digital Imaging and Communications in Medicine, Standard, 2008

[13] Damiani E., di Vimercati S.D.C., Jajodia S., Paraboschi S., Samarati P., Balancing Confidentiality and Efficiency in Untrusted Relational DBMSs, ACM Conference on Computer and Communications Security, 93-102, 2003

[14] Damiani E., di Vimercati S.D.C., Finetti M., Paraboschi S., Samarati P., Jajodia S., Implementation of a Storage Mechanism for Untrusted DBMSs, Proc. of the 2nd IEEE International Security in Storage Workshop, 38-46, 2004

[15] Hacigümüs H., Iyer B., Li C., Mehrotra S., Executing SQL over Encrypted Data in the Database-Service-Provider Model, Proc. of the 2002 ACM SIGMOD International Conference on Management of Data, 216-227, 2002

[16] Yang Y., Ng W., Lau H.L., Cheng J., An Efficient Approach to Support Querying Secure Outsourced XML Information, Conference on Advanced Information Systems Engineering, 157-171, 2006

[17] Jammalamadaka, R.C., Mehrotra S., Querying Encrypted XML Documents, 10th International Database Engineering and Applications Symposium, 129-136, 2006

[18] Lee J.G., Whang K.Y., Secure Query Processing against Encryped XML Data using Query-Aware Decryption, Information Sciences, 176(13), 1928-1947, 2006

[19] Integrating the Healthcare Enterprise (IHE), IHE IT Infrastructure (ITI) Technical Framework 7.0, 2010

[20] U.S. Department of Health & Human Services, HIPAA Administrative Simplification, Federal Register, 2006

[21] Røstad L., Edsberg O., A Study of Access Control for Healthcare Systems Based on Audit Trails from Access Logs, Proceedings of the 22nd Annual Computer Security Applications Conference, 175-186, 2006

[22] Lerner M., Allina Hospitals Fire 32 Over Privacy Violation, StarTribune, May 6, 2011

[23] Abouakil D., Heurix J., Neubauer T., Data Models for the Pseudonymization of DICOM Data, Proc. of the 44th Hawaii International Conference on System Sciences, 1-11, 2011

[24] Neubauer T., Heurix J., A Methodology for the Pseudonymization of Medical Data, Journal of Medical Informatics, 80(3), 190-207, 2011

[25] Heurix J., Neubauer T., Privacy-Preserving Storage and Access of Medical Data through Pseudonymization and Encryption, Proc. of the 8th International Conference on Trust, Privacy and Security in Digital Business, 186-197, 2011

[26] Schrefl M., Dorn J., Grün K., SemCrypt – Ensuring Privacy of Electronic Documents through Semantic-based Encrypted Query Processing, Proc. of the International Workshop on Privacy Data Management, 2005

[27] Grün K., Karlinger M., Schrefl M., Schema-aware Labeling of XML Documents for Efficient Query and Update Processing in SemCrypt, Computer Systems Science and Engineering, 21(1), 65-82, 2006

[28] Shamir A., How to Share a Secret, Communications of the ACM, 22(11), 612-613, 1979