Secure Cloud-based Medical Data Exchange

Christian Neuhaus, Robert Wierschke, Martin von Löwis and Andreas Polze

{christian.neuhaus|robert.wierschke|martin.vonloewis|andreas.polze}@hpi.uni-potsdam.de

Abstract: For decades now, IT technology has been used in the field of medicine and healthcare. Developments in medicine led to a plethora of new diagnostic and imaging possibilities and a flood of corresponding patient data. Nowadays, healthcare is more of an inter-institutional joint effort than it ever was. In IT, the possibilies to process, store and share data have seen a revolution. Both sciences try to keep up with each other, but especially the in the days of ubiquitous networking, ensuring patient data confidentiality has remained a challenge. With cloud computing, a new concept comes along that could fundamentally change the way data is shared and exchanged in healthcare: Low total cost of ownership, excellent scalability and data access without borders of institutions or geographical limitations could provide great benefits. However, the question how to ensure data privacy in such a scenario becomes even more complex. This paper proposes an architecture for a distributed data store based on public cloud storage infrastructures, protected by rights management techniques. The approach is evaluated by showing how it could be applied to the data exchange for the newborn hearing screening programme in Berlin-Brandenburg.

1 Introduction

Computers have been used in medicine and healthcare for a very long time now. Medicine is dependent on accurate information to enable doctors to make decisions and diagnoses [Her02]. Storage, distribution, processing, connecting and presentation of information is, in turn, the prime function of computers. Consequently, a lot has been said, researched and tried out around the question of how computers can encode, interpret, structure, store and retrieve different sorts of medical data. The operating systems research group at Hasso-Plattner-Institut got hands-on experience with data encoding, structuring and transmission of medical data by developing an adaptive communication middleware stack in a telemedicine research project. This research project provides telemonitoring for heart patients in rural areas of northern Brandenburg. These activites result in cooperations with infrastructure providers such as Deutsche Telekom and device manufacturers like GETEMED and BIOTRONIK. Our own publications as well as other related work are listed in section 2.

With the advent of cloud computing [MG09], the way data is handled in general, changes again. Up until a few years ago, state of the art for data availability and interoperability was to provide a public interface to access the information stored in an institution. With cloud storage and computing resources, data can be stored, shared processed, linked, analyzed and searched on very powerful and scalable infrastructure. The application of

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

cloud computing for healthcare purposes could provide benefits such as lower total-cost-of-ownership ([AFG+09]), better availability of the data all over the world and powerful resources to support clinical research on that data. However, with those advantages comes the fact that extremely privacy-sensitive medical data would be stored on infrastructure that is under the governance and control of a third party.

There is an approach that might allow the use of public cloud infrastructures even for such privacy-sensitive applications as healthcare: the concept of *rights management* [Pet07]. The core idea of this concept is to retain control of how, when and by whom data is used, even after it has been disseminated through arbitrary communication channels. Usage of and access to data is only possible with a valid *license* which serves as the access key.

The main benefit from the concept of information rights management is that granting usage rights on data (i.e. issuing the license) is independent and decoupled from data transmission and dissemination. Consequently, data can be transmitted through untrusted channels and stored on untrusted systems. This enables the use of untrusted public cloud storage infrastructures. Usage licenses are only granted to authorized users at trusted end-user sites (e.g. a doctor's office or a hospital), but never to systems on public cloud infrastructures. The security perimeter in this solution is drawn where data is transferred from untrusted public cloud infrastructures to trusted end user sites (see figure 1).

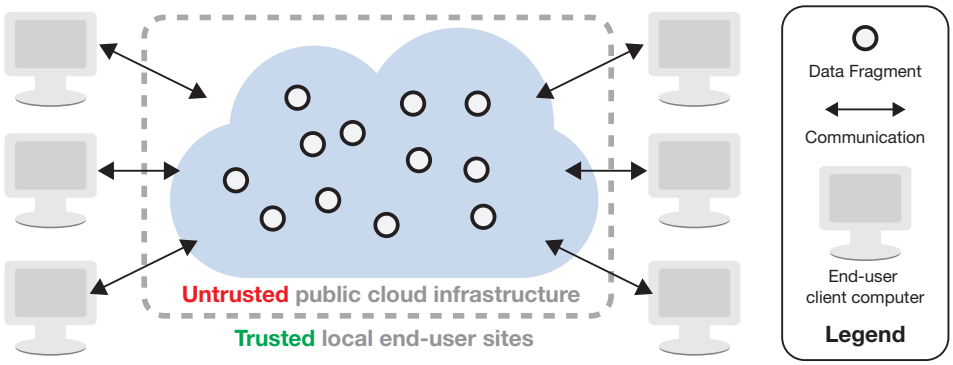

Figure 1: Security Perimeter: Trusted & Untrusted Zones

The contributions of this paper are an architecture and implementation strategies for a distributed cryptographically secured *data store* based on the *information rights management* concept. This datastore can store and retrieve *data fragments*, each accompanied by a security policy and referenced by a unique ID.

We believe that the concept of a *rights management*-protected datastore enhances data privacy and provides a **gain in security**, as it offers a convenient and transparent way of sharing data over a cloud network. The increased security in this approach is due to two reasons: firstly, data is always accompanied by **security metadata**, that shows clearly whom data belongs to and who is allowed to access it. Secondly, data is protected by encryption and is **decrypted as late as possible**, i.e. on end-user systems. Data stays

encrypted during storage and exchange on and over cloud systems. Therefore, cloud infrastructures are only used for storage and exchange, but not for processing of data.

The datastore concept is illustrated in the rest of the paper as follows:

In section 2 we review related work from the areas of security, cloud computing and electronic health records.

In section 3 we present a threat model for the application in a healthcare scenario and list a set of security properties that the datastore should provide.

In section 4 we present the architecture for a middleware software stack and software services that can provide the functionality and security outlined in section 3.

In section 5 we present a scheme for how the functionality of a document-oriented database could be provided on top of the simplistic functionality of a data object store. This scheme is illustrated by an example of how the data exchange of the regional newborn hearing screening programme in Berlin-Brandenburg could be supported this way.

2 Related Work

It has long been recognized that distributed electronic health records and exchange of medical and administrative information between healthcare institutions could provide a great benefit for healthcare both in terms of cost and in quality of care ([Her02], [BG03]). The operating systems group at Hasso-Plattner-Institute is involved in the telemedicine research project Fontane, where an adaptive communication middleware for home monitoring for heart-patients is being developed [PTHH10]. At the same time, electronic storage, processing and exchange of such sensitive information poses a great challenge in maintaining proper privacy and confidentiality of patient data [NCP11].

Different concepts have been proposed to provide privacy in systems that exchange medical information. Encryption and generation schemes for cryptographic keys enable the enforcement of access policies on data items ([ZPJ06], [BCHL09]). Katt et al. [KBH+09] propose an architecture that enables access control for cross-domain document exchange according to policies that are stored in a central repository.

The advent of cloud computing [Pol09] brings a totally new perspective on how medical data can be processed, stored and exchanged. Distributed cloud data storage is researched by [DHJ+07] and [KBC+00]. For security considerations, [PSM09] and [MP09] propose a hybrid approach where data can be processed on cloud systems without revealing sensitive information. To achieve this, sensitive parts of data are cryptographically protected (”obfuscated”), by using Yao’s protocol for secure two-party computation [Yao86]). Results computed on obfuscated data are ”de-obfuscated” on the client-side to reveal the result of the computation. Yao’s protocol requires communication between client and cloud application during computation. This could become obsolete when using homomorphic encryption [VDGHV10].

A general healthcare telematics architecture and a broad spectrum of tools, methods and building blocks for secure exchange of medical data have been developed for the german "Gesundheitskarte" project [FII05]. Probably closest to our work is the research of Kamara et al. [KL10]. They outline requirements and benefits of a secure storage service based on cloud computing. They introduce basic building blocks such as data processors, token generators, credential generators and data verifiers and show high-level architecture blueprints for different scenarios.

3 Concept and Models

This section examines the security threats relevant to a cloud-based data store used for exchange of medical data, states functional requirements and security goals of the system. We describe the abstraction level used for applying security policies to data items. Furthermore, a grammar is provided for the description of security settings for data fragments.

3.1 Security Goal and Threat Model

Our security goal for the data store is to provide *data secrecy*. In this scenario, data is exchanged between end-user systems and cloud storage infrastructures over network connections. Threats to secrecy exist on all three levels. Possible attackers are *curious cloud infrastructure providers*, *curious network providers* and *curious end users*. To presume interest in exchanged medical data in all three of these groups is reasonable: The motivations for sneaking a peek range from human curiosity to concrete monetary interest, such as when insurance companies try to learn about medical conditions of potential customers to adjust their pricing.

In the context of this paper, we do not consider malicious or accidental modification or deletion of stored data, as it is easily detectable by applying digital signatures and checksums to data.

3.2 Requirements for Security and Functionality

As our goal is to provide data secrecy, we want to fulfil the following security requirements when data is distributed over public cloud infrastructures:

Security Parameters: Access to data is granted or denied based on the following parameters:

- **Identity** of the subject requesting access
- **Point in time**

- **Kind of access**: read/write

Fine granularity: Security settings should be fine-granular and applicable on arbitrarily small pieces of data.

Revokable privileges: It is possible to revoke access privileges.[1]

Traceable access: Access to data items is traceable in form of a security audit log.

For the purposes of easier organisation, an *identity* should be constituted by either natural person or a group.

As a functional requirement, the system should provide access to a globally distributed key-value-store, referred to as *data store* in this paper. The value of a key-value-pair will be referred to as *(data) fragment* and should hold an arbitrary byte sequence. The *keys* used to reference and retrieve the fragments are universally unique identifiers. The access functions to manage data fragments should encompass **C**reate-**R**ead-**U**pdate-**D**elete (CRUD) functionality. The data store operates on public cloud storage infrastructures.

3.3 Security Abstraction Level & Data Objects

In the approach of this paper, the object of finest granularity that security settings can be applied to is a data fragment that is saved to the data store. For every data fragment, access rights are specified in its own security policy. The security policy is always distributed alongside the data fragment in the sense of a *sticky policy*[MPB03].

This level of abstraction was chosen because it offers a simple security model based on the simplistic, yet versatile basis of a key-value-store. Key-value-stores conform to the basic structure of a graph where values represent the nodes and edges can be implemented by nodes referencing each other by their key. This basic data structure can be used to represent virtually any other high-level data structure. How more complex functionality can be realized on top of a key-value-store is exemplified in section 5.

Secondly, the scheme of attaching security policies to values deliberately makes no assumption about or requirement on the content of the stored data fragment or its semantic. This does not limit the approach to data content with a certain structure or schema. In turn, the decision which data is to be protected in which way (i.e. what policy to assign to it) is delegated to the application software layer that uses this security infrastructure. This effectively separates security considerations from questions of medical data standards.

The requirement of fine granularity for security settings has one implication for stored data: When different security settings should apply to different parts of a piece of data, the data has to be split into separate data fragments to which different security policies can be applied.

[1] It is not feasible to prevent usage of data that was extracted from the system by reading access. Future writing access and reading access of previously unread data are in fact preventable.

3.4 Security Grammar

For the sake of a simple and comprehensible security model, changes to a security policy are possible to anyone who possesses write privileges to the data fragment. Therefore, write access to a data fragment is a powerful privilege and should be handled with great care.

The access rights in section 3.2 can be specified in a security policy using the following grammar:

Listing 1: EBNF grammar for security policies

```
POLICY = ID_ASSIGNMENT* DATA_OWNER_STATEMENT AC_STATEMENT+;
ID_ASSIGNMENT = IDENTITY "=" <uuid>";";
DATA_OWNER_STATEMENT = "dataowner" IDENTITY";";
AC_STATEMENT = "grant" PRIVILEGE "to" IDENTITY [TIMEFRAME]";"
TIMEFRAME = "within" TIMESTAMP "to" TIMESTAMP;
PRIVILEGE = "read" | "readwrite";
IDENTITY = letter+;
TIMESTAMP = ? iso 8601 timestamp ?;
```

The keyword `IDENTITY` is used where one or more identities as unique identifiers should be provided. For purposes of simplification, an `IDENTITY` can be constituted by either a natural person, a group of people that share a common attribute or an organization.

The `DATA_OWNER_STATEMENT` is used to specify who owns the data contained in the data fragment. This `IDENTITY` always possesses `readwrite` privileges. This way, fragments cannot be ”orphaned” by a data creator locking himself out with an ill-specified security policy.

To illustrate the use of this policy format, an example security policy is given for a fictional data fragment that contains the examination result of a newborn hearing screening:

Listing 2: Example Security Policy for a Hearing Screening Examination Result

```
screeningDoctor  = 9b6fbc5a-3ecc-4dec-876e-e72b299b3557;
andreaMusterfrau = d1e38cd4-66cc-4696-a11a-7b6b090806a4;
erikaMusterfrau  = 9c9396ac-5d69-4dbd-9bce-07822126f8e8;

dataowner screeningDoctor;
grant readwrite to screeningDoctor;
grant read to andreaMusterfrau;
grant read to screeningcenter within 2011-04-28 to 2012-01-01;
```

4 Implementation

This chapter presents a blueprint for the implementation of a rights-management-protected data store that can limit data access to trusted end-user systems (see figure 1) and authorized users and fulfills the requirements made in section 3. An architecture is presented consisting of different cloud-based web-services and user software that can provide the desired functionality and the enforcement mechanisms for security policies are explained.

4.1 Data Store API

Access to the secure distributed data store is provided through a client-side software library which supports Create-Read-Update-Delete-funtionality on *data fragments*. The API can be characterized by the following operations:

Listing 3: Data Store Operations

```
public UUID create(byte[] data, String policy);
public UUID update(byte[] data, String policy, UUID uuid);
public byte[] read(UUID uuid);
public void delete(UUID uuid);
```

The first two operations realize the *create* and *update* functionality. If no UUID is given, the data fragment is created under a new UUID, which is returned. Using `update`, the fragment stored under this UUID is updated. In both cases, a security policy has to be specified in `policy` which conforms to the grammar given in section 3.4. The operation `read` retrieves the fragment with the given UUID from the data store. `delete` removes the fragment with the given UUID from the data store. Generally, the *read*, *update* and *delete* operations are only carried out if the authenticated user has corresponding read/write permissions on the fragment.

4.2 System Architecture

Figure 2 presents a simplified overview over the architecture of the system. It consists of the following building blocks:

Storage Service The *storage service* is a web service running on a public untrusted cloud infrastructure and uses public cloud storage services. It reads and writes data fragments from/to the storage infrastructure and delivers/receives the data to/from the *Client Library* on end-user systems.

License Server The license server evaluates security policies associated to data fragments and provides the requester with usage licences (i.e. decryption keys) if allowed by the policy. The license server is equipped with a public-private-keypair, the public key is publicly known.

Client Library The client library is a software library that is run on every end-user system that needs access to the data store. Therefore, many instances of this software library run in on different systems in different locations. It is equipped with the end-users credentials (public-private keypair) and handles communication with the *storage service* and the *license server*.

Additionally, the system requires a means for lookup and verification of identities, since the access decisions of the license server depend on the identity of the license requester. This could be a user directory that provides public keys for identity verification. This building block of the system is not shown in the diagram.

ISBN 978-3-88579-286-4

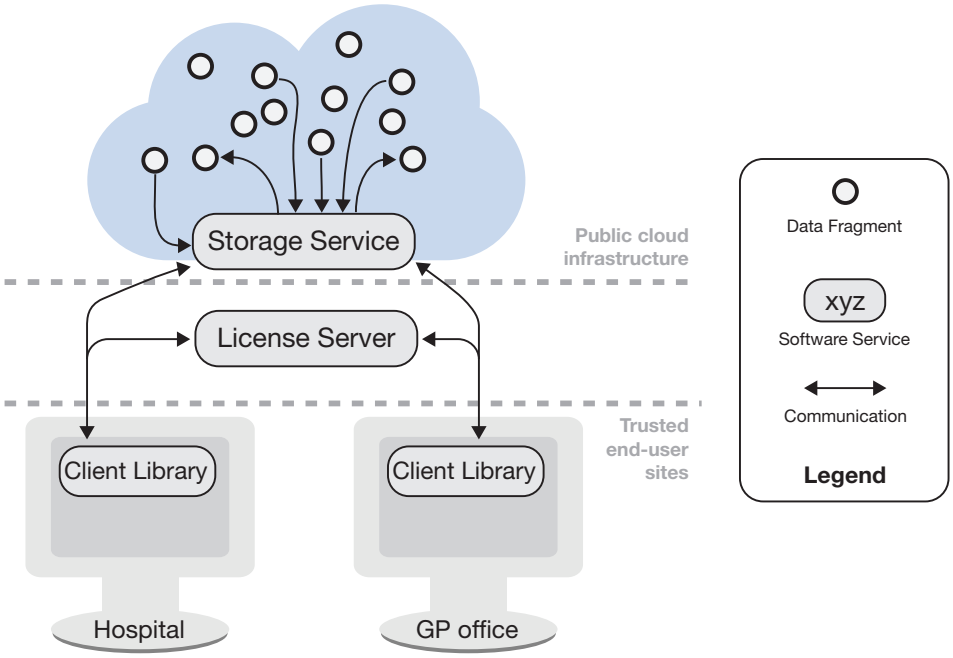

Figure 2: System architecture: Components & Communication

The *license server* and *user directory* services do not necessarily have to be implemented as centralized web services. They merely have to be available to every client library and have to provide coherent answers when queried by the *Client Library*. This could also be realized by providing end-user systems with hardware crypto-devices and a static list of relevant public keys of other users.

4.3 Policy Enforcement

The access rights specified in the security policy of each fragment are enforced in the following way:

The security policy of each fragment is distributed alongside this fragment in the sense of a sticky security policy[MPB03]. This policy is cryptographically bound to the data fragment so data contents cannot be used with a different policy.

Reading access to data fragment is protected by symmetrically encrypting each data fragment with a random symmetric key. This symmetric fragment key is in turn encrypted with the public key of the license server. The encrypted fragment key can only be decrypted with the private key of the license server. The license server decrypts this fragment key on request by users with corresponding rights in the policy.

Write access (update and delete functionality) to a data fragment is enforced by the *storage service*. To do this, the storage service checks whether the requesting end-user possesses write privileges on the data fragment.

4.4 Access Tracing & Privilege Revocation

The requirements on the system demand that accesses to data fragments are traceable and privileges can be revoked. Both requirements are satisfied by usage of a license server, as this service is involved in every access to a data fragment by design. Therefore, an access log of granted licenses can be recorded by the license server. The license server can also be instructed to block access to a particular data fragment by maintaining a blacklist, effectively revoking access privileges this way.

5 Medical Data in a Document Store

Sections 3 and 4 explain in detail the concepts and API of our secure cloud-based data store. In this section we show how to leverage the simplistic data store in order to provide the more powerful functionality of a document store that can be used for storing and exchanging medical data. Furthermore, we demonstrate how it might be used for newborn hearing screening.

5.1 Document-oriented Information Model

The data store presented in section 3 and 4 is simple and flexible. However, an application must have additional knowledge of the structure of the fragments in order to process them and infer relations to other fragments. Furthermore, it is necessary to split the application data into smaller data units (fragments) to be distributed, protected and processed independently. Otherwise, the advantages of a cloud-based solution would be neglected if larger fragments had to be downloaded for processing.

A common approach would be to structure the information using a relational data model [Cod70]. However, the information items that shall be stored and accessed by different users over the cloud must be distinct entities within the system. If a traditional RDBMS was used, the data would be structured in tables: e.g. a table for patient master data and a table for screening data. Thus, the information that is relevant for different users are the rows of such tables or fields within a row. In order to enable fine grained access control on this information, each row needs to be assigned a security policy. Furthermore, each row must be stored and encrypted as its own entity (i.e. fragment).

We suggest to utilize a document-oriented structure (*document store*) for the information model in order to achieve the required information granularity. Here, a document repre-

ISBN 978-3-88579-286-4

sents a combination of logically related key-value-pairs and will be stored as a data fragment in the datastore. The access privileges for every document are specified as a security policy (see section 3.4) and are assigned to the corresponding data fragment in the data store.

5.2 Document Store on top of the Data Store

To implement a document store on the basis of the data store presented in this paper, the following challenges have to be met:

Mapping documents to data store fragments Every document is mapped to exactly one data fragment in the data store. To do this, documents can be serialized using the JavaScript object notation (JSON) [Cro06].

Linking between documents To provide links between the documents, a documents may reference other documents by embedding links. This link is represented by special value similar to the `DBRef` document used in MongoDB [CD10]. A fragment reference is represented by `{"#ref": "<UUID>"}` where `"<UUID>"` identifies the data store UUID of the corresponding data fragment.

Providing a search mechanism A mechanism for searchings documents that contain certain information is required (e.g. retrieving all documents containing patient master data). Since the documents are stored encrypted on the public cloud infrastructure, implementing a search on that infrastructure is difficult. To circumvent the lack of search capability, we manage a set of indices referencing all documents that match certain search criteria. The indices are modeled as document as well. All indices are referenced by a globally known `master_index` document. Thus, assuming matching access right, each index can be reached by following the references from the master index. The structure of the index documents is shown in listing 4. Here `<ref>` represents a reference to an external fragment as described above.

Listing 4: Index Documents

```
master_index := {index1: <ref>, index2: <ref>, ...}
index := {fragments: [<ref>, <ref>, ...]}
```

Proving security granularity To provide fine granularity for security setting, each piece of information with unique security settings has to be stored in its own fragment in the data store, so a corresponding security policy can be applied to it. Consequently, a document may have to be split up into smaller fragments, to which different security policies can be applied. These fragments can be connected by references as described above.

As an example, consider a patients master data in an electronic health record. A document might contain the name and the address of a patient. The address shall not be visible for a nurse involved in the patient's treatment. Hence, the address must be stored in an external document.

ISBN 978-3-88579-286-4

5.3 Case Study: Newborn Hearing Screening

Since January 2009, the hearing of every newborn in Germany is to be tested in the course of the newborn routine checkups[2]. The newborn hearing screening program in Berlin-Brandenburg[3] is a good example where secure cloud-based data distribution is of high value.

The following parties may be involved in such a program:

Screening Site The doctor's office or hospital, where the hearing test is made.

Screening Centre collects the screening data and examines the results for validity

Newborn Preventive Care Centre collects messages of preventive check-ups that have taken place. If parents miss these, an invitation letter is sent to them.

Quality Assurance Centre collects anonymous statistical information of screenings. Quality assurance is required by law to ensure the effectiveness and reach of the screening program.

In order to share the screening data between those parties, the document structure shown in listing 5 could be used.

Listing 5: Document Structure for Screening

```
patient_master_document := {master_data: <ref>, screening: <ref>}
master_data := {pateintid: ..., familyName: ..., givenName: ...,
    birthdate: ..., address: <ref>}
address := {street:..., city:..., postalcode:...}
screening := {screeningID:..., notification:... , result:..., timestamp
    :...}
```

The `patient_master_document` acts as entry point for all information of a patient and contains references to other documents. The data is split up into different documents which contain the actual information or references to other documents for data structuring or security reasons.

While the proposed system allows the sharing of the screening data, not all data should be available to all involved. The patient should be able to view his or her own data. Obviously, the screening site knows the patient's identity as well as the examination result. The quality assurance centre however should only view anonymized examination results. The preventive care centre must know the patients identity but not the examination result. The security requirements are summarized in table 1.

According to the policies from the table, the newborn preventive care centre has access to the `screening` document but not the `result` field. Hence, the `result` field must be protected by encrypting it as an external fragment. In order to achieve all security requirements listed in the table, the `screeningID` as well as the `result` must be encrypted as external fragments.

[2] As enacted by *Gemeinsamer Bundesauschuss* on June 19th 2008, published in *Bundesanzeiger*
[3] http://screening.charite.de/neugeborenen_screening/

	master data	screening			
	*	screeningID	notification	result	timestamp
screening site	x	x	x	x	x
screening centre		x	x	x	x
newborn preventive care centre	x	x	x		x
quality assurance centre			x	x	x
Patient	x	x	x	x	x

Table 1: Access Policy for Screening

Furthermore, in order to allow the different institutions to access the data additional indices are necessary. The patient may access his complete data which can be achieved with a reference to his `patient_master_document`.The quality assurance centre, the newborn preventive care centre, screening centre as well as screening site need an index pointing to the `screening` documents of all patients. Additionally, the screening site, screening centre and the newborn preventive care centre need an index for the patients master data.

6 Conclusion

In this paper, we describe a rights-management protected data object store that enables distribution of sensitive medical information over public cloud storage infrastructures by using *rights management* techniques.

The contributions of this paper are as follows: We provided a threat model for usage of cloud computing infrastructure for medical data. We presented a list of security requirements that a have to be fulfilled by the proposed middleware, and derived a security grammar from those requirements to be able to specify security requirements on data. We outlined the security concept of the system and presented an architecture that can fulfil the specified security requirements and enforce security policies in the specified grammar. For an evaluation, we describe how the more complex data structures of a document store can be mapped to the data object store and show how the data exchange of the newborn hearing screening programme in Berlin-Brandenburg could be carried out over such a system.

Future Work We plan to create a working prototypical implementation of our concept in the near future. The prototype will be developed to reflect the data exchange processes of the hearing screening programme and use realistic patient data.

Our approach provides confidentiality for cloud based data exchange, but does not yet consider availability and reliability of the used infrastructure and data integrity. These aspects should be addressed in the future, building on existing research in the fields of system reliability [AFG+09] and availability [BJO09].

Additionally, proper ways have to be devised how the search indices described in section 5 can be created and maintained. Publish-Subscribe patterns based on fragment metadata might be used for this.

7 Acknowledgements

We would like to thank Günther Laudahn and the company *GN Otometrics GMbH & Co. KG*[4] for lending us one of their hearing screening devices and the corresponding software so that we could gain hands-on experience with realistic data from the hearing screening scenario. We are also grateful to *Deutsche Forschungsgemeintschaft* (DFG) and the SOAMED research school for funding and support.

References

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28*, 2009.

[BCHL09] J. Benaloh, M. Chase, E. Horvitz, and K. Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 103–114. ACM, 2009.

[BG03] D.W. Bates and A.A. Gawande. Improving safety with information technology. *New England Journal of Medicine*, 348(25):2526, 2003.

[BJO09] K.D. Bowers, A. Juels, and A. Oprea. HAIL: A high-availability and integrity layer for cloud storage. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 187–198. ACM, 2009.

[CD10] K. Chodorow and M. Dirolf. *MongoDB: The Definitive Guide*. O'Reilly Media, Inc., 2010.

[Cod70] E. F Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13:377–387, June 1970. ACM ID: 362685.

[Cro06] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). RFC 4627 (Informational), July 2006.

[DHJ+07] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, 41(6):205–220, 2007.

[FII05] SIT Fraunhofer Institutes ISST, IAO. Spezifikation der Lösungsarchitektur zur Umsetzung der Anwendungen der elektronischen Gesundheitskarte. Technical report, Fraunhofer Gesellschaft, 2005.

[4] http://www.gnotometrics.de/

[Her02] W.R. Hersh. Medical informatics: improving health care through information. *Jama*, 288(16):1955, 2002.

[KBC+00] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, et al. Oceanstore: An architecture for global-scale persistent storage. *ACM SIGARCH Computer Architecture News*, 28(5):190–201, 2000.

[KBH+09] B. Katt, R. Breu, M. Hafner, T. Schabetsberger, R. Mair, and F. Wozak. Privacy and Access Control for IHE-Based Systems. *Electronic Healthcare*, pages 145–153, 2009.

[KL10] S. Kamara and K. Lauter. Cryptographic cloud storage. *Financial Cryptography and Data Security*, pages 136–149, 2010.

[MG09] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6), 2009.

[MP09] M. Mowbray and S. Pearson. A client-based privacy manager for cloud computing. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, pages 1–8. ACM, 2009.

[MPB03] M.C. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, pages 377–382. IEEE, 2003.

[NCP11] C. Neuhaus, M. Chowdhury, and A. Polze. Survey on healthcare IT systems: standards, regulations and security. Technical report, Hasso-Plattner-Institute for Software Engineering, 2011.

[Pet07] Milan Petković. Rights Management Technologies: A Good Choice for Securing Electronic Health Records? In *ISSE/SECURE 2007: securing electronic business processes: highlights of the Information Security Solutions Europe/SECURE 2007 Conference*, page 178. Springer, 2007.

[Pol09] Andreas Polze. A Comparative Analysis of Cloud Computing Environments. Technical report, Hasso-Plattner-Institute for Software Engineering, 2009.

[PSM09] S. Pearson, Y. Shen, and M. Mowbray. A privacy manager for cloud computing. *Cloud Computing*, pages 90–106, 2009.

[PTHH10] A. Polze, P. Tröger, U. Hentschel, and T. Heinze. A scalable, self-adaptive architecture for remote patient monitoring. In *2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 204–210. IEEE, 2010.

[VDGHV10] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. *Advances in Cryptology–EUROCRYPT 2010*, pages 24–43, 2010.

[Yao86] A.C.C. Yao. How to generate and exchange secrets. In *— 27th Annual Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

[ZPJ06] A. Zych, M. Petkovic, and W. Jonker. Key management method for cryptographically enforced access control. In *Proc. of the 1st Benelux Workshop on Information and System Security, Antwerpen, Belgium*, 2006.

ISBN 978-3-88579-286-4