Implementing Secure Applications in Smart City Clouds Using Microservices

Michel Krämer[a,*], Sven Frese[b], Arjan Kuijper[c]

[a] *Fraunhofer Institute for Computer Graphics Research IGD, 64283 Darmstadt, Germany*
[b] *wetransform GmbH, 64283 Darmstadt, Germany*
[c] *Technische Universität Darmstadt, 64283 Darmstadt, Germany*

Abstract

Smart Cities make use of ICT technology to address the challenges of modern urban management. The cloud provides an efficient and cost-effective platform on which they can manage, store and process data, as well as build applications performing complex computations and analyses. The quickly changing requirements in a Smart City require flexible software architectures that let these applications scale in a distributed environment such as the cloud. Smart Cities have to deal with huge amounts of data including sensitive information about infrastructure and citizens. In order to leverage the benefits of the cloud, in particular in terms of scalability and cost-effectiveness, this data should be stored in a public cloud. However, in such an environment, sensitive data needs to be encrypted to prevent unauthorized access.

In this paper, we present a software architecture design that can be used as a template for the implementation of Smart City applications. The design is based on the microservice architectural style, which provides properties that help make Smart City applications scalable and flexible. In addition, we present a hybrid approach to securing sensitive data in the cloud. Our architecture design combines a public cloud with a trusted private environment. To store data in a cost-effective manner in the public cloud, we encrypt metadata items with CP-ABE (Ciphertext-Policy Attribute-Based Encryption) and actual Smart City

*Corresponding author
Email address: michel.kraemer@igd.fraunhofer.de (Michel Krämer)

data with symmetric encryption. This approach allows data to be shared across multiple administrations and makes efficient use of cloud resources.

We show the applicability of our design by implementing a web-based application for urban risk management. We evaluate our architecture based on qualitative criteria, benchmark the performance of our security approach, and discuss it regarding honest-but-curious cloud providers as well as attackers trying to access user data through eavesdropping. Our findings indicate that the microservice architectural style fits the requirements of scalable Smart City applications while the proposed security approach helps prevent unauthorized access.

Keywords: Cloud Computing, Software Architecture, Urban Management, Security, Geospatial Information Systems

1. Introduction

Intelligent and effective urban planning is becoming increasingly important. According to a United Nations study on World Urbanization in 2014, more than 50% of the world's population has been living in cities [1]. This number will keep increasing. The United Nations estimate that by 2050 about 66% of the world's population will be urban. Managing large modern cities is already a complex task today but will be even more so in the future.

For this reason, many city administrations have introduced methods and technologies helping them analyze and understand the complex socio-economic interactions in their city and derive plans that facilitate sustainable urban development. Information and communication technology (ICT) is becoming increasingly important for these cities as it allows them to analyze large cross-thematic data (*Big Data*) and to derive knowledge that can be used in the urban development process [2]. The technologies are supported by specialized user interfaces that are easy to understand for domain experts such as urban planners, architects and other stakeholders [3, 4].

A city that makes use of ICT to improve urban development processes is often referred to as a *Smart City* [5]. In other words, Smart Cities make use of intelligent technology to do smart things (or to derive knowledge helping them to do smart things). To achieve their goal of a sustainable and livable urban environment, they perform analyses on large amounts of data. For example, continuously monitoring traffic density, air quality, and weather helps them produce plans to counteract negative implications of recurring traffic jams and their impact on the environment [6]. Laser Mobile Mapping Systems (LMMS) [7] can be used to collect large 3D point clouds in an urban environment, which can be analyzed to monitor tree growth and to foresee pruning works or to plan measures for environmental protection [8]. The data volume necessary for these analyses often exceeds the capabilities of the hardware available to city administrations. At the same time, data and analysis results typically need to be shared among various parties (e.g. departments in a municipality) [9]. For these reasons, Smart Cities make use of cloud technology as it fits most of their requirements with respect to scalability, availability and resilience (see Section 2.1).

Modern approaches to software architecture design such as the microservice architectural style [10] allow for the creation of flexible and scalable software solutions running in the cloud. Microservices are small distributed programs, each of them serving one specific purpose. At the same time, they are highly scalable and resilient. They can be used flexibly to build customized user interfaces for different domain groups (e.g. urban planners, decision makers, or citizens). Their scalability and resiliency go in line with the requirements put on constantly growing cities that must be able to quickly adapt to changes (as depicted in Section 2.1.4). In previous work, we were able to show that microservices can be used to process large amounts of geospatial data in the cloud for various purposes including land monitoring and urban planning [11].

A service-oriented architecture based on microservices therefore fits well in the Smart City domain. However, the data these services have to store and process is often sensitive and requires protection against unauthorized access.

This includes, for example, personal data (names, addresses, birth dates, and the like) as well as information about properties, infrastructure, and criminal offenses. In particular, combining the large amounts of data available in a Smart City network can give deep insight into the personal lives of citizens and can reveal other highly sensitive information. The use case depicted in Section 5.1 deals with estimating the risk of terrorist attacks and calculating consequences and possible counter-measures. In the wrong hands, this information can be dangerous. The services dealing with this data therefore need special protection.

In this paper, we present concepts for building secure applications in Smart City clouds using microservices. We describe an approach based on a hybrid cloud architecture where data is stored in a public cloud, but sensitive processing is performed by microservices deployed to a trusted environment. We encrypt data before it is transferred between microservices and when it is stored in the cloud. The purpose of using public storage over private storage is to have a cost-effective and highly scalable storage solution that supports data sharing between different applications. The encryption scheme we propose utilizes Attribute-Based Encryption (ABE) to support data sharing between users and applications. We use ABE to encrypt small metadata items that are directly accessible. Besides other information, these metadata items contain a symmetric key that is used to encrypt the actual Smart City data. This larger data can be kept in an inexpensive cloud service such as an object store.

In order to demonstrate the benefits of our approach, we apply it to a use case from the urban planning domain. In Section 5.1, we describe an application that can be used to calculate the risk of terrorist attacks and possible damage. Such an application and, in particular, the data it deals with need special protection against unauthorized access. We show that our approach is implementable and can be applied to such a use case.

1.1. Contribution

Although microservices have many benefits regarding flexibility and scalability, they can also lead to increased effort in designing, implementing, and

maintaining a software application, in particular if it needs to be secure as in our case. Since each microservice is a separate program, depending on how the services are deployed and with what kind of data they have to deal with, each of them has to be made secure independently. This is particularly true for Smart City applications dealing with large and sensitive data. In order to prevent unauthorized access to this data, distributed services have to communicate over secure channels and the data should always be encrypted before it is stored. However, it should still be possible to process it in a distributed way to exploit the possibilities of a Smart City cloud. The processing should be scalable, so that large amounts of data can be managed. To the best of our knowledge, there is no work yet describing concepts how to implement secure microservice applications for Smart Cities in the cloud. Our paper tries to fill this gap.

The main scientific contribution of our work is as follows: In order to reduce the required implementation effort, we present a *generic architecture design based on the microservice architectural style* that can be used as a template for secure cloud-based Smart City applications. Our design employs a hybrid model consisting of a public and a private cloud environment. Sensitive data is stored and processed in the trusted private environment, whereas the rest of the application can run in the public cloud. We show that our architecture design is implementable and can be applied to a practical use case.

The second contribution of our work is related to *secure data storage*. In our architecture design, data is encrypted by combining ABE (Attribute-Based Encryption) with symmetric encryption (e.g. AES). We only encrypt small metadata items with ABE, while the actual Smart City data is encrypted symmetrically. We split the metadata from the actual dataset in order to be able to keep them at different locations. This enables us to use cloud services such as object storage to store very large Smart City data while keeping the smaller metadata items at a location that is more directly accessible (e.g. a database or a fast cloud resource such as an SSD block device or a shared file system). As a result, we are able to utilize cloud resources in a reasonable and cost-effective way to manage large datasets and to share them among different parties.

Our application design covers user authentication and authorization, but also secure data storage, transfer between the public and private cloud components, as well as secure processing. The scheduling and management of data processing tasks are not part of this paper. We discuss our design regarding honest-but-curious cloud providers as well as attackers trying to access user data through eavesdropping (see Section 5.4). An evaluation regarding more advanced attacks such as Cross-Site Scripting and Distributed Denial-of-Service attacks is beyond the scope of this article.

Note that the main focus of this paper is on software applications for clouds that provide a dynamic environment of virtual machines running on commodity hardware. This applies to public clouds offered by commercial vendors such as Amazon Web Services (AWS), Microsoft Azure or the Google Cloud Platform, but also to private clouds operated by Smart Cities themselves. Other infrastructures such as Smart Grids, Internet of Things (IoT), or fog computing are beyond the scope of this work.

1.2. Outline

The remainder of this paper is structured as follows. First, we motivate secure Smart City applications in the cloud by explaining the concepts of Smart Cities and their requirements towards effective use of the cloud, as well as describing why Smart City data needs to be stored securely and what issues this raises in a cloud environment (Section 2).

After this, we present related work on software architectures in the cloud, Smart City clouds and security-related topics. We also compare related approaches to our work (Section 3).

In the paper's main part, we describe our software architecture design and give a detailed description of our components (i.e. the microservices) and the data processing workflow. We also explain our approach to securing sensitive data in the cloud and present details on how users can access the encrypted data (Section 4).

Subsequently, we show the applicability of our design by implementing a web application dealing with risk assessment in urban areas. We also perform a qualitative evaluation of our architecture design and present the results of a benchmark of our security method. Finally, we discuss how our approach can help prevent attackers from accessing sensitive data (Section 5).

The paper concludes with a summary of advantages and disadvantages of our approach (Section 6) and an outlook on future work (Section 7).

2. Requirements and motivation

In this Section, we first show how cloud computing can benefit Smart City administrations (Section 2.1) and then explain the insecurities of public clouds and why we propose a hybrid approach (Section 2.2).

2.1. Smart City clouds

As described above, Smart Cities make use of cloud technology to analyze and share large amounts of Smart City data. When building distributed applications, Smart Cities have requirements related to scalability, resilience and self-management that can be covered by the cloud. According to the US National Institute for Standards and Technology (NIST), a cloud can be described by the following characteristics: *on-demand self service*, *broad network access*, *resource pooling*, *rapid elasticity*, *measured service* [12]. In the following, we describe how these characteristics support the goals of Smart Cities and how the cloud matches their requirements. Note that we use these characteristics later in Section 5.2 to qualitatively evaluate our architecture design.

2.1.1. On-demand self service

The cloud allows consumers (i.e. users of the cloud) to acquire computing resources unilaterally without requiring human interaction on the side of the cloud provider. In the context of Smart Cities, the municipality needs to have full control over the resources offered by the cloud. They need to assign resources to certain tasks and revoke them later if they are not needed anymore. Since

modern cities are highly dynamic and unpredictable events can happen at any time, the municipality needs to be able to assign the resources on their own without the help of the cloud provider. The overhead caused by communicating with a third party slows down processes, whereas high automation as it is offered by the cloud provides the required flexibility.

Interestingly, we can find a similar goal in the concept of Smart Cities. One of the aims is to create independent districts that manage themselves without requiring the municipality to take action. A Smart City looks for sustainable solutions with respect to urban planning in order to achieve long-term independence. The notion of self-managing districts matches the requirement to unilaterally and independently acquire resources without interaction on the side of the cloud provider.

2.1.2. Broad network access

The cloud allows a number of clients and devices (e.g. mobile phones, tablets or desktop PCs) to access its resources through unified interfaces. For a Smart City this aspect is important since the infrastructure in a large municipality is typically quite heterogeneous and consists of many different devices. Urban planners and architects use workstations to plan developments in the city but they also need tablets to evaluate and visualize their plans while they are in the field. This means they also need access to data stored in the cloud from virtually everywhere in the city.

Smart Cities often deploy high-performance fiber networks to facilitate broad network access. A recent example from the city of Bristol in the UK shows how various Smart City applications can be implemented using a city-wide network [13]. Bristol uses this network on the one hand to collect data from devices and sensors distributed all over the city and in the future also from autonomous cars. In addition, they offer high-speed Internet access to local companies and deploy WiFi-enabled lamp posts for their citizens.

2.1.3. Resource pooling

A cloud is typically accessed by a number of users at the same time. The cloud resources are dynamically assigned and reassigned depending on the demand from the individual end-user (e.g. an urban planner using a tablet to visualize smart data). The end-users typically do not know how many resources are currently in use or where they are located.

In the context of Smart Cities, this property can be utilized for distributed storage and processing of data. End-users in the municipality can store data in the cloud and gain access to it from anywhere in the city without ever knowing where the data is physically stored. They can also pass on information to colleagues, decision makers or other stakeholders such as companies or citizens through one channel or medium (i.e. the cloud) that appears to be centralized but is in fact highly distributed.

2.1.4. Rapid elasticity

The cloud is often referred to as being able to offer virtually unlimited resources. New resources can be acquired and released on demand in a short amount of time. This feature allows Smart Cities to react on unpredictable events very quickly. For example, in the use case presented in Section 5.1, we describe that the municipality must be able to react on terrorist threats and to quickly calculate consequences and possible counter-measures such as improving building structures or putting up barriers. In this case, it is of major importance that the cloud is reliable and offers any resources required to perform the necessary tasks.

The concept of *rapid elasticity* can also be applied to the Smart City itself. Due to the high speed in which cities are growing nowadays, a Smart City must be designed to be scalable and resilient (i.e. elastic) so it is able to handle future developments and to keep pace with urban growth.

2.1.5. Measured service

In a cloud environment, the use of resources is optimized based on results from comprehensive monitoring. At the same time, monitoring allows cloud providers to create reports that show the amount of resources used by a client in a specific billing period. The reports are often generated in real-time to offer transparency for both the client and the cloud provider.

Municipalities appreciate the fact that a cloud scales on demand. This means that resources can be acquired when necessary, but unneeded resources can also be released to effectively save money. Typically, this is what differentiates a cloud from on-site IT infrastructure which has to be prepared for every case and cannot be scaled down to save resources.

Just like a cloud provider, a municipality needs to monitor developments in the city to optimize urban plans. A Smart City often also provides transparency for their citizens as they make the plans publicly available and offer the possibility to engage in decisions.

2.1.6. Service models

In addition to the characteristics described above, a cloud typically offers three service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). A Smart City offers services in all three models. They make use of external cloud providers but they also deploy their own fiber networks and smart grids to provide a fast and reliable infrastructure (IaaS). Based on this, they offer services and application programming interfaces (APIs) to interested parties such as companies or even freelance programmers who build small applications that they distribute over the city's app store (PaaS). At the same time, the municipalities share collected information with the public as so-called Open Data. Finally, Smart Cities offer services to citizens and other stakeholders such as apps for mobile devices or web portals for public participation (SaaS).

2.2. Public cloud insecurities

Storing the data in a public cloud might be a reasonable choice due to its cost-effectiveness and scalability. However, the security of public cloud storage cannot always be trusted which becomes obvious when taking a closer look at objectives and risks of cloud providers. As outlined by Nanavati et al., cloud providers often use proprietary infrastructure to keep competitive advantages, and in order not to damage their reputation, it is further very unlikely that cloud providers report all bugs and security-related incidents to the public [14]. Another reason to be careful when working with cloud providers is that there can always be weaknesses with the cloud provider's interface as outlined by Somorovsky et al. [15]. Furthermore, we consider providers of public clouds as *honest-but-curious*, which means they store and process the data without tampering but might try to learn the plaintext of the data.

For these reasons, careful evaluation is necessary when using resources by cloud providers in any kind of application that processes sensitive data. As a consequence, in the remainder of this paper, we differentiate between two environments:

Untrusted A public cloud that is not directly under the control of the user (in our case the Smart City administration) is considered *untrusted*.

Trusted A private cloud that is maintained by the user is considered a *trusted* environment.

To use cost-effective and scalable public cloud storage while not relying on the security of the cloud provider's interfaces, as well as to protect the data from being accessed by the cloud provider itself, we propose a hybrid approach consisting of a trusted and an untrusted environment (see Section 4). We encrypt sensitive data before uploading it to the public cloud (see Section 4.1). Data processing that needs the unencrypted plaintext data is performed in the trusted private cloud.

3. Related work

While the microservice architectural style has already been recognized in the industry as the new state-of-the-art approach to build modern cloud-based applications, the topic is still rather unexplored in science. There are a couple of papers, however, dealing with the experiences from applying the microservice approach to real world problems. For example, Vianden et al. present a reference architecture for Enterprise Measurement Infrastructures (EMIs) as well as two case studies in which they apply this architecture to an EMI monitoring software development and another one collecting risk metrics in IT projects [16]. They argue that classic SOA architectures suffer from centralized integration problems such as the need for a common data schema and related mapping problems. To avoid these problems they divide their EMI into dedicated microservices for measurement, calculation and visualization. Their results look promising and they suggest further long-term field studies.

Villamizar et al. report on a case study they conducted to compare a monolithic architecture to one based on microservices and another one running serverless on AWS Lambda [17]. They implemented an example application using these three different architectures and compared performance and response times, but particularly focused on the costs. They conclude that microservices can help reduce infrastructure costs tremendously but the increased effort of implementing and maintaining an application based on this architectural style have to be considered carefully.

In previous work, we used the microservice architectural style to implement a system for the processing of large geospatial data in the cloud [11]. We were able to show that microservices can be used to create a scalable, modular, and maintainable system. In particular, we could demonstrate that the microservice approach allows independent and distributed teams to collaborate and build a joint software.

A couple of works focus on Smart Cities and how they can leverage the cloud. For example, Krylovskiy et al. use the microservice architectural style to

implement a Smart City Internet of Things (IoT) platform [18]. They argue that the microservice approach helps their interdisciplinary and international team to work independently but collaborate efficiently. They also conclude that the growing number of Open Source tools available to build and deploy microservices to various cloud providers will provide long-term benefits, in particular because they allow them to deploy their platform to many cities and different cloud infrastructures.

Khan et al. present a software architecture for a cloud-based analysis service supporting planning and decision making in future Smart Cities [19]. Their solution is based on well-known cloud technologies, tools and open standards making their architecture easy to be integrated into existing Smart City environments. According to them, Smart Cities can benefit from "big, and often real-time cross-thematic, data collection, processing, integration and sharing through inter-operable services deployed in a cloud environment". However, their architecture does not provide a way to process and store sensitive data in a secure way.

In a subsequent work, Khan et al. therefore present a framework for secure and privacy-aware service provisioning in Smart Cities [20]. Compared to our paper, they specifically focus on the security challenges in Smart Cities, as well as the stakeholders and their requirements towards privacy and trustable services. They do not discuss secure data storage or the microservice architectural style.

In the area of secure data storage in the Cloud, Li et al. present an approach to sharing personal health records [21]. They use Key-Policy Attribute-Based Encryption (KP-ABE) to allow patients to setup fine-grained access control to their health records. They improve existing KP-ABE schemes in order to enable efficient and on-demand user revocation. They evaluate the security of their approach and show that their security measures work in an efficient and scalable way. Li et al. mainly focus on security, but also present a web application to manage patient health records. This is in contrast to our work that focuses more on the software architecture. Li et al. do not discuss microservices. While

they show that their security measures are efficient and scalable they do not discuss the scalability or other qualitative criteria of their overall system.

The same applies to the work by Narayan et al. [22]. They also present a system to manage electronic health records stored on honest-but-curious cloud providers in a secure way. Their goal is to maintain data confidentiality and privacy while still keeping the system scalable and access policies flexible. To achieve this, they combine broadcast ciphertext-policy attribute-based encryption (bABE) with symmetric encryption. Files in their system are stored in a way that health record's metadata is encrypted using bABE. Each file's metadata contains a symmetric key. The health records data itself is encrypted with this symmetric key, enabling eligible users to decrypt the health record's metadata using their bABE private keys. They can then use the decrypted symmetric key to decrypt the health record's content using fast symmetric cryptography. Compared to our work, Narayan et al. do not discuss software architecture at all but focus more on Attribute-Based Cryptography. The combination of bABE with symmetric encryption is similar to our approach, but as we describe in Section 4.1.2, we are able to split metadata from larger actual data and therefore make more efficient use of cloud resources. Narayan et al. also do not include a performance benchmark like we do in Section 5.3. Since they do not aim at Smart City applications, both Li et al. and Narayan et al. focus on challenges very different to ours. Their approaches include support for multiple authorities and key revocation mechanisms and therefore use other encryption schemes as we do.

Bugiel et al. outline a secure cloud architecture that uses two distinct components for different tasks [23]. A trusted cloud performs pre-computations while an untrusted but more powerful commodity cloud is used to actually perform user queries. During setup, the trusted cloud encrypts all data and algorithms that need to be performed using garbled circuits. The encrypted data and algorithms are pushed to the commodity cloud where the encrypted algorithms are applied to the encrypted data. When clients query for data, the trusted cloud retrieves the results from the commodity cloud and verifies them before forward-

ing them to the clients. They claim that in their approach, the cloud does not learn anything about the algorithms and the data it processes apart from an upper bound of the data size. Through result verification, malicious commodity cloud providers can further not modify result data without the trusted cloud detecting it.

Popa et al. [24] aim to enable secure data processing for web applications, especially for scenarios where the server provider or operator cannot be trusted. They use asymmetric client-side data encryption as well as key-chaining for secure data sharing between users and groups. The framework they implement provides means to perform computations such as keyword search on encrypted data without revealing critical information, and it further offers a solution to prevent system operators from transmitting malicious client-side code by using a browser plug-in checking the integrity of transmitted JavaScript code.

Our approach shares the notion of a trusted component as suggested by Popa et al. [24] and Bugiel et al. [23]. However, since the risk assessment calculations our web application needs to perform are rather complex, it is in contrast to their approaches not possible to let an untrusted cloud perform them on encrypted data, for instance by using garbled circuits. Our software architecture is therefore fundamentally different from theirs.

4. Approach

In order to support the implementation of scalable and secure applications, we propose a concept that is based on a hybrid cloud architecture. The system architecture design consists of three major components being a web interface layer, a private processing cloud as well as public cloud storage as depicted in Figure 1.

One of the most important parts in our design is the *trusted component*. It represents a set of services that run in a trusted environment (i.e. the private cloud; see definition in Section 2.2). We describe it in detail in Section 4.2.1.

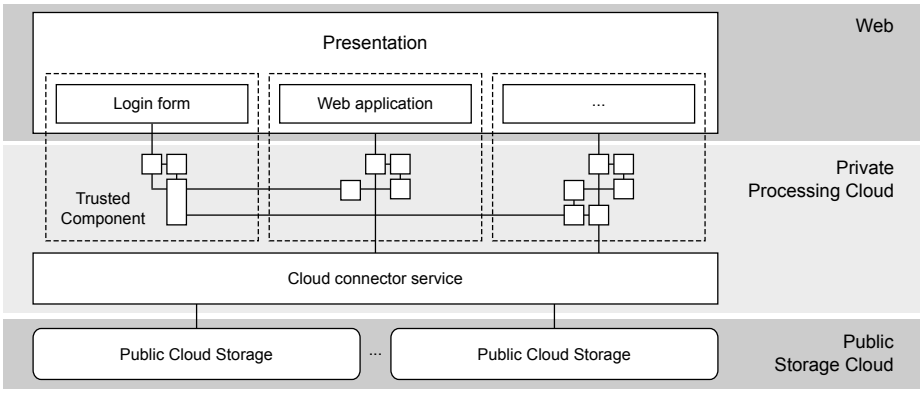

Figure 1: The major components of our system design are a web interface, a cloud for processing private data and public cloud storage.

The microservices are divided into bounded contexts [25] denoted by dashed lines in the Figure. The trusted component and the web application represent two different bounded contexts. In the architecture diagram, we included a third optional and unnamed bounded context suggesting that there may be further system components (or web applications) accessing the trusted component and the data shared in the public cloud.

In order to process data and perform operations that users request via any of the web applications, the web interface layer is connected to the private processing cloud. The private processing cloud consists of all microservices that are required to provide the overall functionality of the applications implemented in the Smart City cloud. In this design, each microservice is implemented to provide distinct functionality. The overall functionality of Smart City cloud applications is then achieved through tight communication and orchestration of the functionalities provided by the individual microservices.

As a solution to store and share all data that is gathered and processed by the Smart City cloud, our system architecture utilizes public cloud storage. The purpose of using public storage over private storage is to have a cost-effective and highly scalable storage solution that supports data sharing between different applications. Since data is stored using external cloud storage providers, it

is important to protect the confidentiality of sensitive data. But at the same time, this data needs to be accessible by all eligible users and shared between applications. To satisfy both requirements, we utilize a hybrid encryption scheme provided by services implemented in the private processing cloud in order to keep data confidential and support access control in a scalable way. In the remainder of this section, we describe our secure storage model in detail before outlining how the private processing cloud is orchestrated in order to support the workflow of different Smart City cloud applications.

4.1. Confidential scalable storage

In order to protect the confidentiality of data stored in the public cloud, all sensitive data needs to be encrypted before being uploaded. To be able to scale these operations and at the same time retain the confidentiality of the processed data, all encryption and decryption operations are performed in the private processing cloud. Our encryption scheme uses Attribute-Based Encryption (ABE) to support data sharing between users and applications. We combine ABE with symmetric encryption to make efficient use of cloud resources.

4.1.1. Attribute-Based Encryption

The main benefit of *Attribute-Based Encryption* (ABE) is that it allows for embedding access policies in the encryption process and enforcing them based on the private key used during decryption. This way, data can be shared amongst a group of users. The concept of ABE has first been introduced by Sahai et al. [26] as a new approach for Identity-Based Encryption [27, 28]. Sahai et al. outline that identities can be derived from a set of attributes each individual possesses. They propose a key generation scheme that uses sets of attributes to generate distinct private keys representing these attributes. In ABE, all parties holding a public key can encrypt data while only users with private keys matching a pre-defined number of attributes can decrypt the ciphertext.

In order to be able to use ABE in a larger variety of systems that require more fine-grained access control, Goyal et al. [29] have developed the concept of *Key-Policy Attribute-Based Encryption* (KP-ABE). In KP-ABE each ciphertext labels attributes that are required to decrypt it. Each private key contains a monotone access structure over these attributes. A party can decrypt a ciphertext if the attributes of the ciphertext are included in its private key's access structure.

In addition to KP-ABE, there is *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) which has been introduced by Waters et al. [30]. CP-ABE follows a different strategy than KP-ABE: instead of storing access policies in the private keys, it is included in the ciphertext. Accordingly, the private keys expose access attributes that the owner possesses. While previous work outlined notions of ABE conceptually and focuses on security concerns, Bethencourt et al. [31] were the first to implement a CP-ABE scheme for practical use and perform extensive performance evaluation. As their performance evaluation reveals, CP-ABE can induce quite significant overhead due to its use of pairing-based cryptography. Its encryption and decryption times increase linearly with the amount of attributes contained in the access structure.

4.1.2. Our approach to secure data storage

We utilize CP-ABE rather than KP-ABE since it allows for more flexibility in data access structures and less overhead when new access attributes are introduced in the overall system. We propose to combine CP-ABE with symmetric encryption (see Figure 2). In the first step, *sensitive data* is encrypted using a symmetric encryption scheme such as AES. For each sensitive data item there is a *metadata item* generated. The metadata item can contain any information on the sensitive data that should stay confidential such as the *author of the data* or the *location where it should be stored*. Additionally, the metadata item contains the *symmetric key* used to encrypt the data item. The metadata item is then encrypted using CP-ABE under an access policy defined by the data owner. With this approach, CP-ABE encryption and decryption only need to be ap-

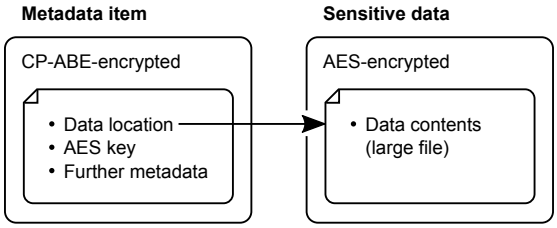

Figure 2: The CP-ABE-encrypted metadata item contains the symmetric key and a pointer to the larger AES-encrypted file body.

plied on the small metadata item while the actual dataset can be encrypted and decrypted using significantly faster symmetric cryptography.

Our approach has a major benefit. The metadata item contains a 'data location' attribute pointing to the actual sensitive data. Due to this, we are able to store metadata item and the larger dataset at separate locations. For example, the metadata item can be kept in a database or another fast cloud resource such as a block device or a shared file system, while the actual dataset can be stored in an inexpensive cloud service such as an object store. Note that both locations can be in the public store as all items are encrypted. As a result, we can make efficient use of cloud resources. We refer to Section 5.3 for a performance benchmark and a comparison to other work.

4.2. Secure data processing

As mentioned in Section 4, our storage solution is supported by a private processing cloud. The benefit from this approach over other approaches such as encrypted processing [23] is the greater flexibility regarding operations that can be performed on the data, as well as significantly better performance since no complex operations need to be performed on encrypted data.

However, our system design comes with the drawback that processing of sensitive data cannot be performed on public cloud servers. Therefore, in addition to public cloud storage, a capable private cloud infrastructure needs to be set up and scaled to the required tasks.

4.2.1. Trusted Component

To perform secure data processing and support the encryption operations required by our storage model, besides other services providing functionality for the applications implemented in the Smart City cloud, the private processing cloud contains a set of services that form a trusted component. The trusted component consists of an *authentication microservice*, a microservice to handle encryption and decryption operations (the *crypto service*), as well as a *private database connector microservice* with a private database (see Figure 3).

The *authentication microservice* is the first service any user or other microservice has to interact with when accessing protected data. The authentication microservice handles authentication and authorization based on accounts which are required to view any sensitive data.

Associated with each account is a secret as well as a private key. While the secret is used for authentication, the private key is generated using CP-ABE

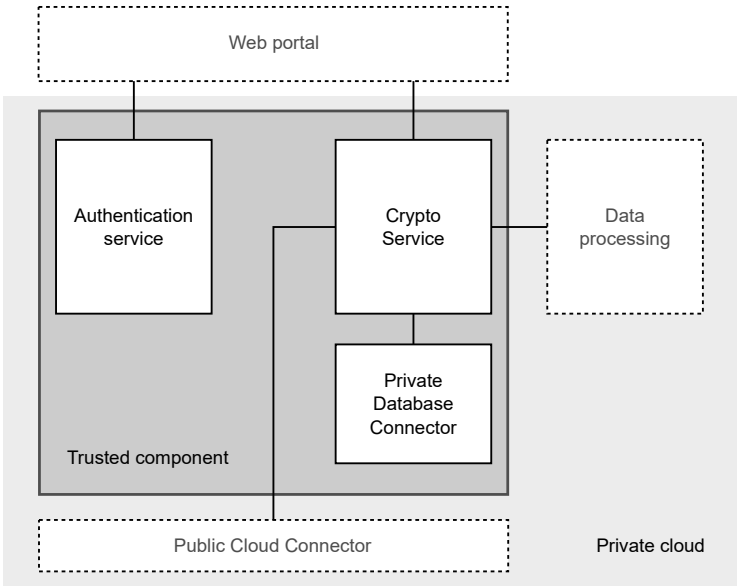

Figure 3: The trusted component consists of three microservices (authentication service, crypto service and private database connector) and runs in the private cloud (detailed view of Figure 1 on page 16)

and holds the access attributes associated to the account. In order to protect account data from being accessed by any unauthorized service from the private processing cloud, it is stored in a local private database which can only be queried through the *private database connector microservice* from within the trusted component.

The third service of the trusted component is the *crypto service*. It performs all encryption and decryption operations required in order for other services in the private processing cloud to be able to process and store sensitive data. The operations supported by this microservice especially encompass encryption and decryption with CP-ABE as well as symmetric cryptography with AES.

4.2.2. Data processing workflow

Having introduced the tasks performed by the services forming the trusted component, the next step is to show how other services implemented in the private processing cloud work together and use the trusted component in order to store and process sensitive data in a secure way. To show the typical data flow in our system design, we use the example of an application capable of performing risk assessment for urban areas based on geospatial and empirical data that has been implemented based on our approach (see Section 5.1).

Account generation and authentication. In order to access any resource provided by the Smart City cloud, urban planners need user accounts. While we're not discussing account generation strategies as part of this paper, our approach assumes that after successful account generation the following information is available to the authentication service:

- **ID:** A unique user ID.
- **Email:** The user's email address.
- **Password:** The hashed user password.
- **Applications:** A list of Smart City cloud applications the user is allowed to access.

- **Private Key:** The private key of the user, generated with CP-ABE under the user's access attributes. The key is encrypted using the user's password.

After successful account creation the urban planner can access the application through its web interface. To be able to use the application, the urban planner needs to authenticate first. In order to handle authentication and authorization for all available Smart City cloud applications, our system implements a *Single Sign-On* (SSO) authorization system based on OAuth2. This enables users to authenticate only once and use all applications associated with their user accounts without requiring them to enter their credentials each time they access a different application.

When accessing the first application, the user's browser is redirected to a login page provided by the authentication service where the user credentials are entered and submitted to the authentication service along with a URL the browser should redirect to after successful authentication. The authentication service verifies the user credentials by comparing the hashed password provided by the user to the hashed user password stored in the private account database. Using the redirection URL, the authentication service also verifies the authenticity of the web application the user has requested login from. It is checked whether the application is a trusted application implemented in the Smart City cloud and whether the user has access to this particular application based on the list of allowed applications stored upon account creation.

Upon success, the authentication service issues an access token indicating that the user is allowed to access Smart City cloud resources using the web application that initiated the login. Additionally, the authentication service uses the password provided by the user in order to decrypt the private key associated with the user account. The decrypted key is stored only for the length of the user's session along with the current access token and the session expiration time.

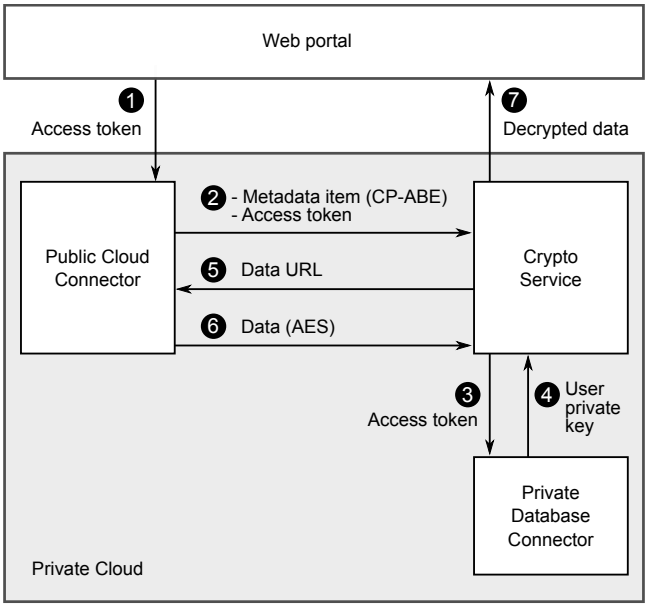

Figure 4: The secure data retrieval process of a web application implemented in the Smart City cloud.

After successful login, the browser of the user is redirected to the front page of the risk assessment web application. In the process, the user's access token is passed to the application.

Secure data access from applications. In order for the risk assessment application to actually load risk assessment data for the user currently logged in, the previously transmitted access token is used as shown in Figure 4. The web portal sends a request containing the user's access token for the metadata items of the data to be loaded which is forwarded to a cloud storage connector service ❶. The service then retrieves the encrypted metadata from the public cloud storage. In order to present the user with decrypted data, the metadata is then forwarded to the crypto service along with the access token ❷. Since the crypto service is part of the trusted component (see section 4.2.1), it has access to the user's account information. In order to decrypt the CP-ABE encrypted metadata item, the crypto service requires the user's private key. To retrieve it,

it queries the local database connector, which is also part of the trusted component, using the access token ❸. The local database connector replies with the decrypted private key of the user if the user's session is still active ❹. The crypto service then attempts to decrypt the encrypted metadata item with the private key. If the access attributes of the user embedded in the key match the access policy embedded in the ciphertext of the metadata item, the decryption will succeed. If the user's private key doesn't match the access policy required to access the encrypted data, decrypting the metadata item will fail and the user won't be granted access to the requested data. In case the user's private key does match the set access policy and decrypting the metadata item succeeds, the decrypted metadata is being used to query the public cloud storage connector again for the actual data ❺. The actual data is then passed to the crypto service for symmetric decryption with the symmetric key contained in the previously decrypted metadata item ❻. The service performs the decryption and replies with the decrypted data which upon success is then forwarded to the web portal ❼.

If risk assessment data has been created or altered through the web portal, the results need to be properly encrypted before they can be securely stored in the public cloud. To store the data securely in the public cloud, the web portal needs to pass the data to the crypto services that generates a symmetric key and encrypts the data with it. The service replies with the encrypted data along with the symmetric key. Then a new metadata item is being created which is then passed to the crypto service along with the desired access policy. The crypto service uses CP-ABE in order to encrypt the metadata item under the given policy before passing the result to the public cloud connector in order to store the symmetrically encrypted data as well as the CP-ABE encrypted metadata item.

4.3. Interfaces

To prevent unauthorized access to protected data from other parties than the public cloud provider, our Smart City cloud architecture needs to properly

protect its interfaces. As we have already outlined, our architecture implements an SSO approach in order to properly authenticate users before authorizing them to access protected data. To prevent common Man-in-the Middle attacks aiming to learn user secrets, all communication to and from the web interface uses the HTTPS protocol and is therefore encrypted with TLS/SSL.

This does not only protect the user's credentials during authentication, but also the access token from unauthorized access. The token is always required when an operation provided by the web interface of the Smart City cloud is being accessed. The web service to be invoked passes it to the authentication service with every incoming request in order to verify that the user the access token has been issued to actually has access to the queried web service.

Inside the Smart City cloud, all calls to the trusted component also require passing the user's access token in order to, for instance, decrypt protected data retrieved from the public cloud storage connector. Except for the described functionalities to verify access tokens and request decryption of user data, the trusted component does not provide any further services to other microservices implemented in the private processing cloud in order to protect the user account data from being accessed or altered by other microservices.

5. Evaluation

The main contribution of this paper is the software architecture design presented in Section 4. In the following, we first show the feasibility of our design based on an example implementation of a real-world use case (Section 5.1). We further evaluate our software architecture based on qualitative criteria (Section 5.2) and measure the performance of our encryption scheme (Section 5.3). Finally, we critically discuss the security of our approach (Section 5.4).

5.1. Implementation

In this Section, we describe a web application that is designed to support urban planners in the process of identifying and ultimately reducing security

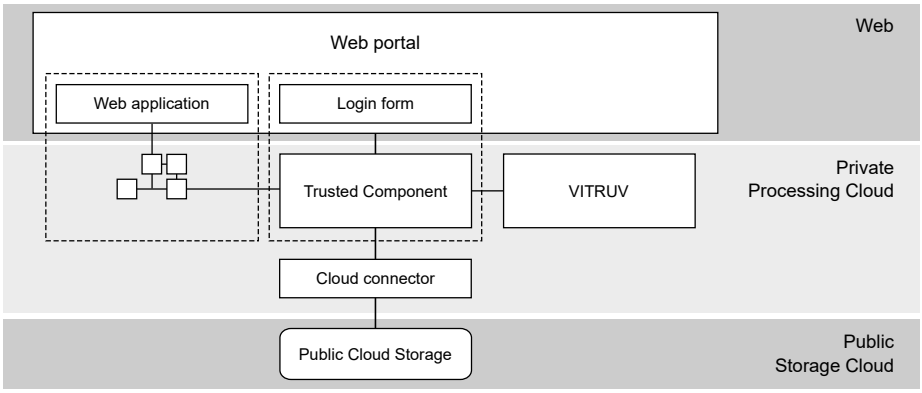

Figure 5: The components of our example application and how the are deployed to a private and a public cloud

risks in urban areas. It makes use of the VITRUV Tool which is a software performing risk assessment based on digital city models as well as historical data [32]. The VITRUV Tool creates a visualization indicating, for instance, the vulnerabilities and susceptibilities of buildings with regards to actions with criminal intent such as burglaries or even bombings. The VITRUV Tool provides many possibilities for municipalities and other authorities concerned with urban security. First, its risk assessment results can be used to identify existing vulnerable spots in urban centers to, for instance, mitigate consequences of terrorist attacks. Another use case of the VITRUV Tool is in the process of planning of new structures, in particular when considering their security. Risk assessment results can be used to evaluate the best spots for new potentially endangered buildings such as embassies. A third use case is the evaluation of safe spots for upcoming public events such as speeches, demonstrations and festivals.

Because the risk assessment calculations performed by the VITRUV Tool are quite complex, they should not be run on desktop computers but rather dispatched to scalable cloud resources. This however leads to the problem that sensitive data needs to be stored and processed on untrusted cloud resources. Since it is possible that parties with malicious intent might also be interested

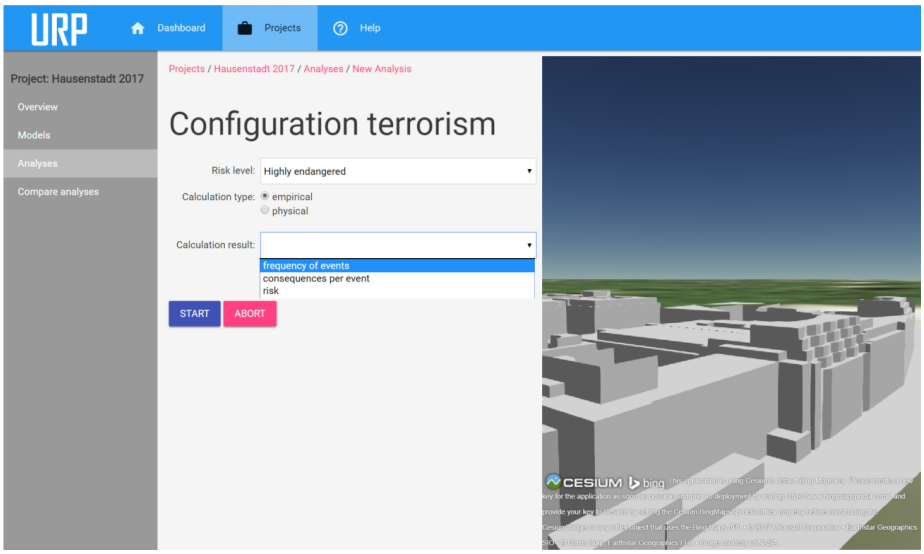

Figure 6: Screenshot of our web application for urban risk assessment

in information on building vulnerabilities, all data that is stored and processed by such a cloud-based application needs to remain confidential at all times.

In order to achieve both scalable calculations and secure data processing, this web application has been implemented using the system design we have presented in this paper (see Section 4 and in particular Figure 1). Figure 5 shows the architecture of our application and how its microservices are deployed to a private and a public cloud. The web application consists of a user-interface running in a web browser and a set of microservices. The functionalities provided by the VITRUV Tool are also wrapped in a microservice that can be queried using HTTPS. In addition, there is the trusted component consisting of authentication service, crypto service and private database connector (individual services not shown in the Figure, see Section 4.2.1).

All of these microservices are deployed in the private processing cloud. We implemented the services with Vert.x [33], a toolkit for building reactive applications on the Java Virtual Machine. The crypto service utilizes the libbswabe library by Bethencourt et al. [31] for CP-ABE encryption and decryption operations. The overall web application functionality can be accessed via an HTTPS

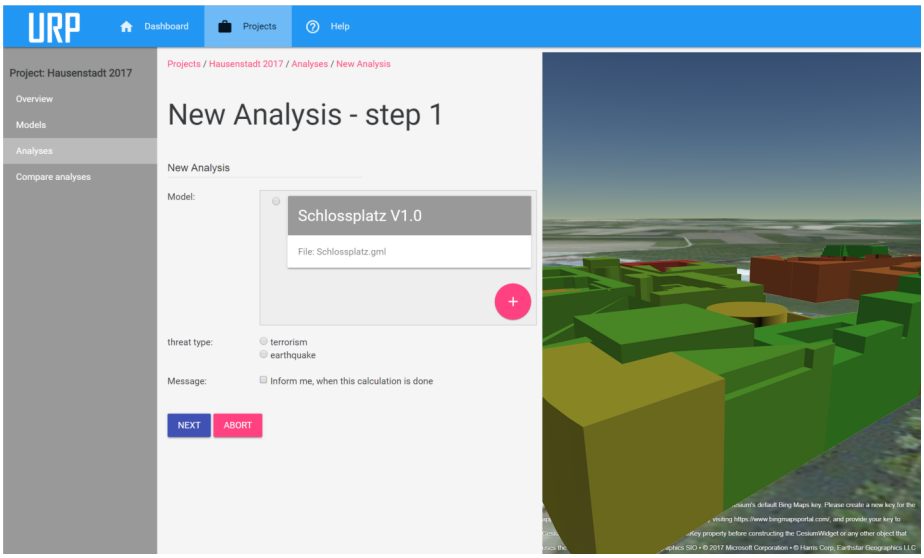

Figure 7: Our web application with 3D buildings colored according to the result of the risk assessment

interface again provided by a microservice. In order to protect sensitive data also in transfer to and from the user, all external communication to and from the HTTP interface is protected through TLS/SSL. The microservices communicate with each other over secure channels.

Figures 6 and 7 show screenshots of our final web application. Whenever urban planners want to start a risk assessment calculation, they need to upload a digital city model first. The uploaded data is encrypted before it is being forwarded to the cloud storage. A subset of the uploaded data is sent over a secure channel to the VITRUV service. Once the risk assessment is finished, the results are fetched from the VITRUV service and stored securely in the public cloud.

The urban planner is kept updated about the risk assessment's status by our web application and can request the results through the web portal once they are ready. Accessing the result data from the web portal then works as outlined in Section 4.2.2. Finally, the data is converted to a 3D scene that can be

visualized in the web browser. We use the Open Source framework Cesium [34] to visualize geospatial data through WebGL.

5.2. Qualitative evaluation

After showing the applicability of our architecture design in the previous section we now evaluate it conceptually. For this, we go back to the requirements of Smart Cities described in Section 2.1 and show how they can be mapped to the properties of our architecture.

Smart Cities have a need for *on-demand self service* (Section 2.1.1). On the one hand, this requirement is satisfied by the cloud that allows users to unilaterally acquire computing resources. On the other hand, this only applies to the service models offered by the cloud provider (often only IaaS and PaaS) but not to the software itself. Our architecture design represents a template to create secure applications that allow end-users such as urban planners or other members of the municipality to leverage the possibilities of the cloud without having to care about technical details or how to keep their data secure. As shown in Section 5.1, it is possible to implement an application where users can assess urban risks on-demand and unilaterally.

In this respect, we note that the microservice architectural style allows for creating highly available applications that are tolerant to faults. Individual microservices can be deployed redundantly so that if one instance fails, another one can take over. The Vert.x framework we used in our implemented application offers a high-availability feature with automatic fail-over. This is necessary to implement unilaterally on-demand self service.

The requirements *broad network access* (Section 2.1.2) and *resource pooling* (Section 2.1.3) are generally satisfied by the cloud and not by our architecture design per se. However, we can combine them with our architecture's scalability to achieve the other requirement *rapid elasticity* (Section 2.1.4). Our design allows individual microservices to be deployed redundantly, not only for fault tolerance as mentioned above, but also to increase performance and throughput. If the number of instances of a service scales with the number of users (i.e.

resource pooling), we can implement *rapid elasticity* by deploying a load balancer that distributes requests to these instances. Due to our encryption scheme and the *broad network access* offered by the cloud, we can process large amounts of data with a reasonable performance (see Section 5.3).

Our architecture does not include specific monitoring and billing services that, for example, measure how many times the VITRUV service has been used or how much data was encrypted. Such services are beyond the scope of this paper. Regarding the requirement *measured service*, we therefore need to rely on the mechanisms provided by the cloud providers.

In summary, the microservice approach and our architecture design fit quite well to the concepts of a Smart City. They allow for better scalability and modularity facilitating simplified development and a shorter time-to-market. Individual (small) services can be developed and provided to citizens and interested parties in a short amount of time. Outdated services can be removed or replaced quickly by new versions. All of this supports the aim of Smart Cities to create a constantly evolving environment worth living in.

As described earlier in this paper, the public cloud offers many benefits to Smart Cities, in particular in terms of scalability, performance, and cost-effectiveness. However, in our architecture security-related computations have to be done in a private cloud which has to be maintained by the city itself or by a trusted company that is subject to the same legislation as the city. Managing an own cloud infrastructure takes efforts and can be costly for the city. Nevertheless, we expect that the infrastructure needed for our trusted component can be relatively small compared to the larger public cloud. Computations on non-sensitive data can still be performed in the public cloud. Due to the fact that our architecture is based on microservices, most parts of the web application can be put in the public cloud as long as they do not deal with sensitive data. This enables the Smart City to leverage the benefits of the public cloud while keeping the sensitive parts of the application in a protected environment.

5.3. Encryption performance

In this Section, we present the results from a benchmark we performed to test the performance of our encryption scheme. In our approach, we encrypt small metadata items with CP-ABE and actual Smart City data symmetrically. We compare this to plain CP-ABE applied to the data as a whole. As explained in Section 5.1, we use the CP-ABE implementation libbswabe by Bethencourt et al. [31].

For the benchmark, we set up an environment consisting of the a *private OpenStack cloud* operated by our research institute and *Amazon Web Services (AWS)*. We deployed our trusted component to the OpenStack cloud and configured the public cloud connector to use the object store *Amazon S3* for large Smart City datasets and the database *Amazon DynamoDB* for metadata items. Since our private cloud was located in Darmstadt, Germany, we chose the closest AWS region 'eu-central-1' in Frankfurt, Germany.

The test data was a subset of an open dataset provided by the German federal state of North Rhine-Westphalia "Land NRW (2018)" licensed under the dl-de/by-2-0 (Datenlizenz Deutschland - Namensnennung - Version 2.0, `www.govdata.de/dl-de/by-2-0`) available at `https://www.opengeodata.nrw.de/produkte/geobasis/3d-gm/3d-gm_lod2/`. The file "3d-gm_lod2_05315000_Köln_EPSG25832_CityGML.zip" contained the 3D city model of Cologne divided into multiple tiles in the CityGML format (Level of Detail 2). It had a size of 4 GB. We selected 100 representative files from this archive with a total size of 2,088 MB. The individual file sizes ranged from 3.6 MB to 50 MB. The average file size was 20.37 MB. The median was 21.5 MB.

We ran the following five tests for each approach (ours and plain CP-ABE):

1. *Encrypt and upload the dataset.* We assigned CP-ABE policies to the encrypted files alternating between an attribute a_1 and another one a_2. This means that 50 out of 100 files required a_1 to be present in the user's attribute set and the other 50 required a_2.

2. *Download and decrypt with a_1.* We created a user possessing a_1 and then tried to download and decrypt all files. With a_1 it should be possible to decrypt 50 files only.
3. *Download and decrypt with a_2.* We created another user possessing a_2. This user should be able to decrypt the other 50 files.
4. *Download and decrypt with both attributes.* The third user possessed a_1 and a_2 and should be able to download all files.
5. *Download and decrypt without attributes.* The fourth user had no attributes and should not be able to access any files.

We measured the times for encryption and decryption (both CP-ABE and symmetric AES), the upload and download times, as well as the number of bytes transferred during upload and download of the actual 3D city model and the metadata. Note that the benchmark was executed inside the trusted component, so all numbers for upload and downloads times as well as the data sizes relate to data transfer between our private cloud and the public cloud. We did not examine the communication between our web application and the trusted component since the performance would be the same regardless of the chosen approach.

Table 8 shows the numbers recorded during upload. As we can see transferred data sizes are the same, but our approach also needs to upload the additional metadata items to DynamoDB. However, the overhead is very small compared to the size of the actual dataset and does not affect the (rounded) sum of transferred data sizes. The sum of upload times for data and metadata in our approach almost matches that of plain CP-ABE (with small fluctuations in the upload rate). The sum of the encryption times with CP-ABE (for metadata) and AES (for the actual data) in our approach is lower than that of plain CP-ABE. The small difference is in line with the observations made by Bethencourt et al. [31].

Tables 9 and 10 show the results from downloading the data with our approach an plain CP-ABE respectively. Again, our decryption takes less time

	Our approach	Plain CP-ABE
AES encryption time	7.0 s	–
CP-ABE encryption time	2.3 s	16.4 s
Data upload time	112.9 s	118.9 s
Metadata upload time	2.9 s	–
Data upload size	2,088 MB	2,088 MB
Metadata upload size	97.6 KB	–
$\sum$ encryption times	9.3 s	16.4 s
$\sum$ upload times	115.8 s	118.9 s
$\sum$ upload sizes	2,088 MB	2,088 MB

Figure 8: Benchmark results from uploading the dataset with our approach and plain CP-ABE

than plain CP-ABE. The main difference is the number of bytes downloaded. The tables show that, with our approach, a lot less data needs to be downloaded and the overall process is therefore much faster. With plain CP-ABE, every file has to be downloaded from the public cloud before its policy can be validated. With our approach, we can save bandwidth by only downloading those files that have a policy matching the user's attributes. We still need to download all metadata items, but this overhead is very small compared to the size of the whole dataset.

In summary, our approach is faster than plain CP-ABE in all cases. Even if we have to download the whole dataset, our encryption and decryption take less time.

In addition, our encrypted metadata items can hold more information than just the symmetric key. We can also store the dataset's author (or owner), the creation date, quality attributes, or anything else that could be relevant for the secure Smart City application. The application has therefore direct access to the stored metadata attributes and can use them without downloading the whole dataset from the object store.

	a_1	a_2	a_1 and a_2	(none)
AES decryption time	1.0 s	1.2 s	1.8 s	0.0 s
CP-ABE decryption time	1.1 s	1.1 s	1.3 s	0.8 s
Data download time	22.4 s	19.3 s	42.1 s	0.0 s
Metadata download time	2.9 s	3.1 s	4.8 s	2.0 s
Data download size	1,045 MB	1,043 MB	2,088 MB	0 MB
Metadata download size	97.6 KB	97.6 KB	97.6 KB	97.6 KB
$\sum$ decryption times	2.1 s	2.3 s	3.1 s	0.8 s
$\sum$ download times	25.3 s	22.4 s	46.9 s	2.0 s
$\sum$ download sizes	1,045 MB	1,043 MB	2,088 MB	97.6 KB

Figure 9: Benchmark results from downloading the dataset with our approach by different users possessing a_1, a_2, a_1 and a_2, as well as no attributes (download rates fluctuated slightly)

	a_1	a_2	a_1 and a_2	(none)
CP-ABE decoding time	8.0 s	7.8 s	13.9 s	1.5 s
Data download time	49.4 s	44.0 s	49.3 s	43.7 s
Data download size	2,088 MB	2,088 MB	2,088 MB	2,088 MB

Figure 10: Benchmark results from downloading the dataset with plain CP-ABE (download rates fluctuated slightly)

5.4. Security discussion

We now discuss aspects related to the introduced security measures regarding honest-but-curious cloud providers as well as attackers trying to access user data through eavesdropping.

Data in our system can be stored in two ways. Sensitive user information is stored within the trusted component. This data is only utilized inside the trusted component and can therefore not be leaked to eavesdropping attackers or curious cloud providers. Data stored using public cloud storage are CP-ABE-encrypted metadata items as well as the AES-encrypted data items. They can not be accessed by eavesdropping attackers and cloud providers since keys to decrypt the data are only being utilized in the trusted component. The

only information cloud providers can gather in our system design are relations between metadata items and actual data items by tracking access patterns to both kind of files.

To keep sensitive data private, we encrypt it using CP-ABE with symmetric encryption. However, when data is encrypted, sharing it can become an issue. In our scenario, it is quite likely that multiple urban planners from one or even multiple municipalities need to share data in order to perform different risk assessment calculations and share experience regarding effective security measures. Sharing encrypted data often involves finding solutions for managing and sharing multiple keys which can lead to new issues regarding data security. In CP-ABE, however, we only have to protect one private key per user that contains their access policies, whereas access policies are safely embedded in metadata items pointing to the actual encrypted files. We also do not need any further keys for managing organizations since that information would be part of the user's attribute set that is used to generate the private key. Not using organization keys also saves us a significant amount of encryption operations and storage since in such an approach files would also have to be encrypted multiple times with each organization key [24]. It would also make it difficult to share data between organizations and single users, whereas in our approach, the metadata item of the file to be stored just have to be re-encrypted with an access policy that also matches the attributes of users to share the data with. However, if the access attributes of a user change, for instance when switching organizations, we just have to generate a new private key for that user reflecting the new set of access attributes. Since the key is handled by our trusted component, a complicated revocation mechanism is not required for such a case.

The major drawback of using CP-ABE is the performance overhead it introduces. As the performance study by Bethencourt et al. [31] shows, its performance depends highly on the complexity of the access policy of the encrypted files. This means if complex access policies are required to enable sharing data between municipalities and users, the encryption and decryption processes of

shared files will slow down the overall system. However, since with our approach, we only have to encrypt small metadata items instead of the actual files that need to be shared, this overhead is still moderate.

6. Conclusions

In this article, we presented a system architecture to protect data in Smart City clouds. We first discussed the motivation of Smart Cities to use the cloud and the security issues that may result from that. We then compared related work to our approach and presented our architecture design as well as the security measures we apply. Finally, we presented results from implementing a real-world application for urban risk assessment based on our design. We evaluated our architecture quantitatively based on the requirements of Smart Cities. We also measured the performance of our encryption scheme and critically discussed the advantages and disadvantages of our approach.

With the implementation, we were able to show that our architecture design fits well to a real-world Smart City application. Our approach to securing data in the cloud covers many security aspects and can be employed flexibly. The microservice architectural style opens a number of possibilities to integrate our system design into existing infrastructures. Our services have lightweight interfaces and communicate over HTTP. Each individual service has a manageable size and serves a specific purpose. This enables better maintainability, a shorter time-to-market, as well as improved flexibility and scalability.

Combining a private cloud with the public cloud enables Smart Cities to protect sensitive data while leveraging the possibilities of a distributed environment at the same time. The public cloud offers many benefits (as described in section 2.1) but can become a security risk if sensitive data needs to be processed or stored. The private cloud, on the other side, is a safe environment for any data that needs protection against unauthorized access. Managing a private cloud in addition to the public one means further effort and costs, but

since the major part of data a Smart City deals with is non-sensitive the private infrastructure can be considerably smaller than the public one.

As our performance evaluation has shown, our approach to encrypt data and to separate metadata items from the actual Smart City data is faster than plain CP-ABE in all cases. In most cases, the number of bytes that need to be transferred is much smaller so that our approach is significantly faster than the existing one.

The encryption scheme we applied in our approach protects sensitive data against honest-but-curious cloud providers as well as attackers trying to access it through eavesdropping. In particular, in our example application, CP-ABE has proven to be quite useful because it allows for sharing datasets and computation results between different parties—in our case multiple departments of the same municipality.

7. Future Work

In order to further improve data confidentiality in a real-world scenario, it remains as future work to evaluate the security of our approach and our developed web application against active attacks on the system. Common attacks that should be considered in the evaluation include Man-in-the-Middle attacks, Cross-Site Scripting and DDoS attacks.

In addition, it would be worthwhile investigating data integrity approaches. Our system design only keeps data confidential but does not provide means to test if the protected data has been modified or is still complete. In our web application, this would help make sure users receive valid risk assessment results at all times.

In parallel to this work, we also investigated the use of implementing encryption directly in the data store while still allowing operations such as importing, deleting and—most importantly—querying without leaking information about the query itself and its results [35]. We extended the geospatial data store

GeoRocket [36] with dynamic searchable symmetric encryption. Combining this approach with CP-ABE as presented in this paper is still open future work.

We will continue our work on the microservice architectural style and how it can be applied to the cloud paradigm. In previous work we have presented a workflow management solution for the processing of large geospatial data in the cloud [37, 11]. We implemented the workflow manager as well as the individual processing algorithms as microservices to achieve best scalability and fault tolerance.

The same applies to our work on microservices in the context of Smart Cities. We are currently working on a platform for participatory urban governance within the EC-funded project smarticipate [38]. This platform is based on microservices and runs in the cloud. The security scheme presented in this work could be integrated in this platform to protect sensitive data about citizens or the urban infrastructure.

We did not investigate the use of our approach for other network infrastructures such as Smart Grids, Internet of Things (IoT), or fog computing yet. Our approach has been specifically designed for web applications running in dynamic cloud environments such as Amazon Web Services (AWS), Microsoft Azure, the Google Cloud Platform, as well as private clouds.

References

[1] United Nations, World urbanization prospects: The 2014 revision, `http://esa.un.org/unpd/wup/Highlights/WUP2014-Highlights.pdf`, [Online; accessed 27-Feb-2018] (2014).

[2] Z. Khan, A. Anjum, K. Soomro, M. A. Tahir, Towards cloud based big data analytics for smart future cities, Journal of Cloud Computing 4 (1) (2015) 2. `doi:10.1186/s13677-015-0026-8`.

[3] M. Krämer, D. Ludlow, Z. Khan, Domain-specific languages for agile urban policy modelling, in: W. Rekdalsbakken, R. Bye, H. Zhang (Eds.),

Proceedings of the 27th European Conference on Modelling and Simulation (ECMS), European Council for Modelling and Simulation, Ålesund, Norway, 2013, pp. 673–680.

[4] J. Dambruch, M. Krämer, Leveraging public participation in urban planning with 3D web technology, in: Proceedings of the Nineteenth International ACM Conference on 3D Web Technologies (Web3D), Web3D '14, ACM, New York, NY, USA, 2014, pp. 117–124. doi:10.1145/2628588.2628591.

[5] D. Ludlow, Z. Khan, Participatory democracy and the governance of smart cities, in: 26th Annual AESOP Congress, Ankara, Turkey, 11th - 15th July, 2012.

[6] D. H. Hoang, T. Strufe, Q. D. Le, P. T. Bui, T. N. Pham, N. T. Thai, T. D. Le, I. Schweizer, Processing and visualizing traffic pollution data in Hanoi City from a wireless sensor network, in: 38th Annual IEEE Conference on Local Computer Networks - Workshops, 2013, pp. 48–55. doi:10.1109/LCNW.2013.6758497.

[7] B. Sirmacek, R. Lindenbergh, Automatic classification of trees from laser scanning point clouds, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences II-3/W5 (2015) 137–144. doi:10.5194/isprsannals-II-3-W5-137-2015.

[8] J. Böhm, M. Bredif, T. Gierlinger, M. Krämer, R. Lindenbergh, K. Liu, F. Michel, B. Sirmacek, The IQmulus Urban Showcase: Automatic Tree Classification and Identification in Huge Mobile Mapping Point Clouds, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLI-B3 (2016) 301–307. doi:10.5194/isprs-archives-XLI-B3-301-2016.

[9] C. Yang, M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus, D. Fay, Spatial cloud computing: how can the geospatial sciences use

and help shape cloud computing?, International Journal of Digital Earth 4 (4) (2011) 305–329. doi:10.1080/17538947.2011.587547.

[10] S. Newman, Building Microservices, 1st Edition, O'Reilly Media, Inc., 2015.

[11] M. Krämer, A microservice architecture for the processing of large geospatial data in the cloud, Ph.D. thesis, Technische Universität Darmstadt (2018). doi:10.13140/RG.2.2.30034.66248.

[12] P. M. Mell, T. Grance, SP 800-145. The NIST Definition of Cloud Computing, Tech. rep., National Institute of Standards & Technology, Gaithersburg, MD, United States (2011).

[13] J. Temperton, Bristol is making a smart city for actual humans, http://www.wired.co.uk/article/bristol-smart-city, [Online; accessed 20-Feb-2018] (2015).

[14] M. Nanavati, P. Colp, B. Aiello, A. Warfield, Cloud security: A gathering storm, Commun. ACM 57 (5) (2014) 70–79. doi:10.1145/2593686.

[15] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwenk, N. Gruschka, L. Lo Iacono, All your clouds are belong to us: Security analysis of cloud management interfaces, in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW '11, ACM, New York, NY, USA, 2011, pp. 3–14. doi:10.1145/2046660.2046664.

[16] M. Vianden, H. Lichter, A. Steffens, Experience on a microservice-based reference architecture for measurement systems, in: 2014 21st Asia-Pacific Software Engineering Conference, Vol. 1, 2014, pp. 183–190. doi:10.1109/APSEC.2014.37.

[17] M. Villamizar, O. Garcés, L. Ochoa, H. Castro, L. Salamanca, M. Verano, R. Casallas, S. Gil, C. Valencia, A. Zambrano, M. Lang, Infrastructure cost comparison of running web applications in the cloud using aws lambda and

monolithic and microservice architectures, in: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2016, pp. 179–182. doi:10.1109/CCGrid.2016.37.

[18] A. Krylovskiy, M. Jahn, E. Patti, Designing a smart city internet of things platform with microservice architecture, in: 2015 3rd International Conference on Future Internet of Things and Cloud, 2015, pp. 25–30. doi:10.1109/FiCloud.2015.55.

[19] Z. Khan, A. Anjum, S. L. Kiani, Cloud based big data analytics for smart future cities, in: Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC '13, IEEE Computer Society, Washington, DC, USA, 2013, pp. 381–386. doi:10.1109/UCC.2013.77.

[20] Z. Khan, Z. Pervez, A. Ghafoor, Towards cloud based smart cities data security and privacy management, in: IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 806–811. doi:10.1109/UCC.2014.131.

[21] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption, IEEE Trans. Parallel Distrib. Syst. 24 (1) (2013) 131–143. doi:10.1109/TPDS.2012.97.

[22] S. Narayan, M. Gagné, R. Safavi-Naini, Privacy preserving EHR system using attribute-based infrastructure, in: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop, CCSW '10, ACM, New York, NY, USA, 2010, pp. 47–52. doi:10.1145/1866835.1866845.

[23] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, T. Schneider, Twin clouds: Secure cloud computing with low latency, in: Proceedings of the 12th IFIP TC 6/TC 11 International Conference on Communications and Multimedia Security, CMS'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 32–44.

[24] R. A. Popa, E. Stark, S. Valdez, J. Helfer, N. Zeldovich, H. Balakrishnan, Building Web Applications on Top of Encrypted Data Using Mylar, Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14) (2014) 157–172.

[25] Evans, Domain-Driven Design: Tackling Complexity In the Heart of Software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[26] A. Sahai, B. Waters, Fuzzy Identity-Based Encryption, EUROCRYPT 2005, Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (2005) 457–473doi:10.1007/11426639_27.

[27] A. Shamir, Identity-based cryptosystems and signature schemes, in: G. Blakley, D. Chaum (Eds.), Advances in Cryptology, Vol. 196 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1985, pp. 47–53. doi:10.1007/3-540-39568-7_5.

[28] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: J. Kilian (Ed.), Advances in Cryptology – CRYPTO 2001, Vol. 2139 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 213–229. doi:10.1007/3-540-44647-8_13.

[29] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based Encryption for Fine-grained Access Control of Encrypted Data, Proceedings of the 13th ACM Conference on Computer and Communications Security (2006) 89–98doi:10.1145/1180405.1180418.

[30] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: D. Catalano, N. Fazio, R. Gennaro, A. Nicolosi (Eds.), Public Key Cryptography – PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 53–70. doi:10.1007/978-3-642-19379-8_4.

[31] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 321–334. doi:10.1109/SP.2007.11.

[32] Fraunhofer Institute for High-Speed Dynamics, Ernst-Mach-Institut, EMI, VITRUV Tool, http://www.vitruv-tool.eu/, [Online; accessed 28-Feb-2018] (2014).

[33] The Vert.x Project, Vert.x, http://vertx.io/, [Online; accessed 27-Feb-2018] (2017).

[34] Analytical Graphics, Inc. (AGI), Cesium - WebGL Virtual Globe and Map Engine, https://cesiumjs.org/, [Online; accessed 27-Feb-2018] (2017).

[35] B. Hiemenz, M. Krämer, Dynamic searchable symmetric encryption for storing geospatial data in the cloud, International Journal of Information Securitydoi:10.1007/s10207-018-0414-4.

[36] Fraunhofer Institute for Computer Graphics Research IGD, GeoRocket - A high-performance data store for geospatial files, https://georocket.io, [Online; accessed 27-Feb-2018] (2017).

[37] M. Krämer, I. Senner, A modular software architecture for processing of big geospatial data in the cloud, Computers & Graphics 49 (2015) 69–81. doi:10.1016/j.cag.2015.02.005.

[38] The smarticipate project consortium, Smarticipate, https://www.smarticipate.eu/, [Online; accessed 28-Feb-2018] (2017).