A Cloud Design for User-controlled Storage and Processing of Sensor Data

René Hummen*, Martin Henze*, Daniel Catrein†, Klaus Wehrle*
*Communication and Distributed Systems, RWTH Aachen University, Germany
Email: {lastname}@comsys.rwth-aachen.de
†QSC AG, Germany
Email: {firstname.lastname}@qsc.de

***Abstract*—Ubiquitous sensing environments such as sensor networks collect large amounts of data. This data volume is destined to grow even further with the vision of the Internet of Things. Cloud computing promises to elastically store and process such sensor data. As an additional benefit, storage and processing in the Cloud enables the efficient aggregation and analysis of information from different data sources. However, sensor data often contains privacy-relevant or otherwise sensitive information. For current Cloud platforms, the data owner looses control over her data once it enters the Cloud. This imposes adoption barriers due to legal or privacy concerns. Hence, a Cloud design is required that the data owner can trust to handle her sensitive data securely. In this paper, we analyze and define properties that a trusted Cloud design has to fulfill. Based on this analysis, we present the security architecture of SensorCloud. Our proposed security architecture enforces end-to-end data access control by the data owner reaching from the sensor network to the Cloud storage and processing subsystems as well as strict isolation up to the service-level. We evaluate the validity and feasibility of our Cloud design with an analysis of our early prototype. Our results show that our proposed security architecture is a promising extension of today's Cloud offers.**

***Keywords*-Cloud, WSN, Security, Architecture**

I. Introduction

Today, the boundaries between the physical world and the digital world continue to blur due to advances in the areas of ubiquitous computing and wireless sensor networks [1]. At the same time, the *Cloud computing* paradigm has become an established alternative to dedicated on premises data storage and computation resources. Both trends, ubiquitous sensing and Cloud computing, complement each other in a natural way. Sensor networks collect information about the physical environment, but typically lack the resources to store and process the collected data over long periods of time. Cloud computing elastically provides the missing storage and computing resources. Specifically, it allows to store, process, and access the collected sensor data effectively via Cloud-based services. To illustrate this fact, consider the following example: Private weather stations do not only provide a local view on current sensor readings, but additionally transmit their measurements to forecast services running in the Cloud. The Cloud services process and aggregate the received sensor readings and use them in a Cloud-based weather simulation that generates an accurate forecast for a specific region. This forecast is then fed back to the private weather stations in order to provide its owner with an added value service.

Cloud-based services that operate on sensor data would largely contribute to enriching or creating new services such as the one described above. However, sensor data often contains privacy or otherwise sensitive (meta-)information. For example, the weather forecast service may require the sensor data owner to reveal her location in addition to the raw, insensitive temperature values. While the data owner may be willing to share this sensitive information with services that she authorized, she oftentimes does not trust the Cloud provider or other services to handle her sensitive data securely. Therefore, she may refrain completely from using Cloud-services that are based on her sensor data. This dilemma highlights the adoption barriers for Cloud computing on sensor data. These adoption barriers arise due to the loss of control of the data owner over her data as previously identified for related scenarios [2], [3].

There exist a number of approaches that aim at providing secure data storage and computation in the Cloud. These approaches typically focus on providing hard security guarantees by using cryptographic primitives such as trusted platform modules or homomorphic encryption [4], [5], [6]. However, the proposed approaches are inadequate for the purpose of storing sensitive sensor data in today's Cloud environments. They either do not provide the necessary user control over outsourced data in the Cloud or they introduce excessive encryption overhead when applied to comparably small sensor data [7]. Hence, we see the need for a practically viable approach to storing and processing sensor data in the Cloud.

Our contribution is as follows: Firstly, we analyze the specific threats and security requirements that arise when outsourcing storage and processing of possibly sensitive sensor data to the Cloud. Secondly, we discuss how these properties can be assured and communicated to the user in brief. Based on this discussion, we propose the security architecture of SensorCloud. SensorCloud provides a platform (PaaS) for the execution of services that operate on sensor data while satisfying and ensuring the derived trust and security requirements. Our proposed security architecture implements i) early protection of sensor data already in

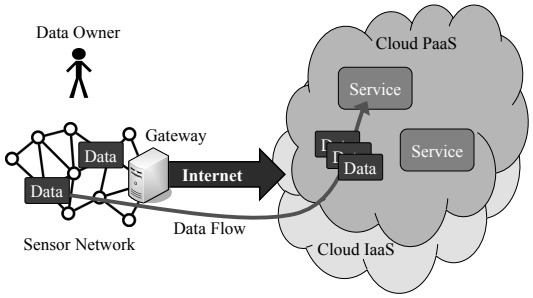


Figure 1. Abstract overview of a generalized Cloud scenario when outsourcing storage and processing of sensor data to the Cloud. Data flows originate from individual sensors and are forwarded by the gateway to the Cloud. Within the Cloud, data traverses the IaaS, PaaS, and SaaS layers.

the sensor network, ii) user-controlled granting of access to user-selected Cloud services, and iii) strict service isolation within the Cloud platform. These properties enable the data owner to stay in control over which entities may access her data in the Cloud and, thus, make Cloud-based services on sensor data viable.

II. SensorCloud scenario and security concerns

Sensor networks typically are dedicated and isolated networks that collect information about their environment. The sensed raw data is commonly collected by a sink node that may either pre-process the raw data on-site or forward it directly to its actual consumer. In either case, the recipient of the collected information is known at the data sink. Hence, the collected data only leaves the network domain of the data owner on distinct and controlled paths.

However, when the data owner outsources storage and processing of sensor data to the Cloud, the paths that this data takes become ambiguous as multiple entities contribute to the overall service provisioning. More precisely, data sent by the sensor gateway towards the Cloud traverses the IaaS and PaaS layers before being processed by a service (see Figure 1). The IaaS layer virtualizes its available physical resources and offers these to multiple IaaS users. The PaaS layer is one of these users. It implements the run-time environments and APIs that grant a multitude of services access to the received and stored sensor data. As a result of this layered architecture and the inherent use of multi-tenancy, potentially sensitive sensor data traverses an unknown set of systems. To overcome adoption barriers arising from this uncertainty, a trusted Cloud design must prevent access to information by the Cloud provider and third parties, unless the data owner explicitly agrees to share it. We now analyze the specific threats when outsourcing sensor data to the Cloud in order to identify the issues that a trusted Cloud design needs to protect against.

A. Threats for Outsourced Data in the Cloud

Due to the involvement of multiple stakeholders, there exists a number of threats that are specific to outsourcing data to the Cloud. To address these threats in a structured way, our analysis follows the high-level security objectives identified by NIST [8]. These objectives are *confidentiality*, *integrity*, *availability*, *accountability*, and *assurance*. We focus our discussion on threats specific to the SensorCloud scenario. Generic threats for Cloud computing (e.g., infrastructure availability) are also relevant. However, these have already been studied extensively in [3], [9], [10], [11].

Data confidentiality and integrity: Unprotected sensor data transmitted to the Cloud for storage and processing can be accessed or altered by entities other than the legitimate stakeholders involved in the respective operations. Data in transit between the sensor network and the Cloud infrastructure may be acquired or deliberately modified by an attacker that is located on or besides the communication path. Furthermore, sensor data in the Cloud is exposed to potential access or modification by privileged staff of the Cloud provider (IaaS or PaaS provider). Finally, Cloud storage and computing resources are typically shared by a multitude of users and services that are unknown to each other. Thus, unauthorized users or services may gain access or modify information if the isolation mechanisms of the Cloud platform are insufficient.

Data accountability: Once sensor data enters the Cloud platform, the data owner loses control over who can access her data. Specifically, the IaaS or PaaS provider (including their employees) may forward stored data to unauthorized third parties without the data owner acknowledging or even recognizing this transfer. Likewise, a Cloud service may pass data that it is authorized to process to a third party unnoticed by the data owner. Consequently, sensitive information may potentially spread across Cloud platform boundaries without the knowledge of the data owner.

Service Availability: The data owner becomes dependent on the SaaS provider when transferring her sensor data to the Cloud for subsequent processing. This is particularly true when the result of the processing is time critical, e.g., to trigger an actor or raise an alarm. Hence, the data owner requires a certain service availability guarantee from the SaaS provider. The SaaS provider, in turn, can only fulfill its guaranteed availability if the PaaS provider can ensure availability of its platform and the IaaS provider can ensure the availability of its physical resources. Thus, the disruption of Cloud service by a malicious Cloud user, e.g., by imposing high load on the shared physical infrastructure, is highly intolerable on any Cloud layer.

Assurance: Assurance denotes the confidence that technical and operational security measures work as intended to meet the objectives discussed above. Both false positive and false negative assurances impose risks. A data owner may falsely conclude that all the required security measures are in place as she is not an expert in the domain of Cloud security and it is beyond her technical capability to verify the measures in detail. As a consequence, while she assumes her data

to be secure, one or more of the threats associated with the objectives discussed above persist. Likewise, the lack of transparency regarding the measures used in the Cloud platform to secure the data storage and processing may effectively cause the platform to be perceived by the data owner as insecure as today's Cloud platforms. As a result, the data owner would consider the overhead of the employed security mechanisms as additional costs. These costs may drive her to cheaper but less secure Cloud competitors.

We now give a brief overview of the principal building blocks that are at our disposal in order to solve the identified threats in a complex system such as a public Cloud platform.

B. Building Blocks of a Trusted Cloud

There exist three high-level instruments that support the stakeholders of a Cloud system in counteracting the threats discussed above: legal agreements, processes, and technology. *Legal agreements* allow the IaaS, PaaS, and SaaS providers as well as the data owner to make binding statements about their service requirements and provisioned service properties. An often-cited example are service level agreements stating technical properties such as a service availability of 99.999%. Furthermore, legal agreements can also refer to processes, e.g., regular certification or audits of the Cloud platform. To ensure the commitment of the contract partners, legal agreements commonly introduce fines when not obeying the agreed obligations.

Processes allow the contract partners to determine and ensure that legal obligations are met. In the context of a trusted Cloud platform, processes can be used by the Cloud provider to prove, e.g., that the technology applied to provide security guarantees is correctly implemented and used. Correspondingly, processes cover the development, operation, and certification or audit of a Cloud platform.

Technology represents the foundation of a trusted Cloud platform. With respect to outsourcing storage and processing of sensor data to the Cloud, technology needs to provide the means to protect the privacy and security of sensitive sensor data in a multi-tenant system outside the trust domain of the data owner. The main technologies used today for this purpose are cryptography and separation of concerns, e.g., via virtualization.

We briefly discuss the role of processes and legal agreements in the following description of our Cloud security architecture. However, due to space restrictions, the main focus of this paper lies on the technological aspects of a trusted Cloud platform that form the underlying basis for processes and legal agreements. We now proceed with the description of our proposed security architecture.

III. SensorCloud Security Architecture

The SensorCloud security architecture emphasizes control of the data owner over her sensor data when outsourcing storage and processing to the Cloud. To enable this control, the SensorCloud security architecture primarily has to meet the following high-level requirements.

Data confidentiality and accountability: The data owner must be able to control who can access her data in- and outside the Cloud platform.

Data integrity: The data owner and Cloud services must be able to verify that the stored data has not been manipulated.

Service availability: The Cloud PaaS provider must ensure that the Cloud IaaS layer is available and that Cloud services are accessible.

With respect to assurance, we assume that the IaaS and PaaS provider ensure compliance to security standards such as the Common Criteria for Information Technology Security Evaluation (CC) or ISO/IEC 27001 by means of independent audits and certifications. These audits and certifications allow to assure the Cloud users that the required security guarantees have been considered and met already since the early development processes of the Cloud platform.

Additionally, we make the following assumption while addressing our high-level requirements. Within the boundaries of the *sensor network*, we assume that data is transferred securely to the gateway according to its sensitivity. This involves confidentiality and integrity protection where necessary. Considering the *gateway*, we assume that its configuration is secure and access control mechanisms are implemented properly. Additionally, the gateway is able to uniquely match incoming sensor data to the corresponding sensing device. With respect to the *Cloud* platform, we assume that the Cloud providers themselves are not hostile. Specifically, the IaaS and PaaS provider operate technology, services, and interfaces as contractually agreed. Furthermore, we assume that the IaaS cloud is reliable and cleanly separates different users. Thus, other IaaS users can neither influence the operation of the PaaS platform nor access data and services while they are processed in the PaaS. Finally, a *Cloud service* that is authorized by the data owner to process her data holds to its appropriation as agreed with the data owner. Most importantly, the Cloud service does not hand data over to an unauthorized third party.

A. Bridging the Sensor Network and Cloud Domains

We reason that the gateway connecting the sensor network with the external network (e.g., the Internet) is the common point of control for all data leaving the data owner's trust domain. Hence, we realize our control and security mechanisms at this border of the sensor network. Specifically, we introduce the *Trust Point* as a new logical entity that is situated on the gateway. It acts as a bridge between the security domain of the sensor network and the Cloud and performs the following three tasks. First, it forwards the sensor data to the Cloud platform on behalf of the data owner. Second, it protects the confidentiality and integrity of the forwarded data during the transmission to the Cloud.

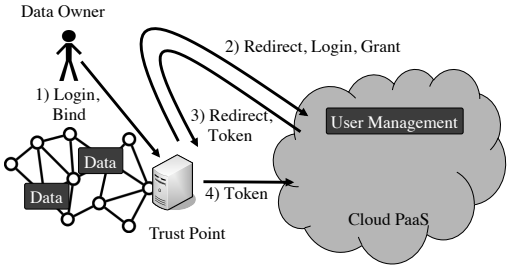


Figure 2. OAuth protocol flow in SensorCloud. The data owner triggers the binding procedure at the Trust Point and is redirected to the Cloud platform. She then authenticates and authorizes the Trust Point. After the redirection to the Trust Point, a secure connection is established between the Trust Point and the Cloud platform.

Third, it applies per-data item security measures in order to establish control of the data owner over her data even after the transport protection is terminated in the Cloud. We discuss these tasks in the following sections.

The PaaS provider offers storage resources for sensor data to the data owner. For reasons of accountability, we require the data owner to register with the Cloud platform. As the Trust Point stores data in the Cloud on behalf of the data owner, we need to bind its identity to the data owner's identity in the Cloud. We do not directly use the owner's credentials (e.g., username and password) on the Trust Point as a data owner may own several Trust Points for different sensor networks. Instead, we identify the Trust Point by means of public key cryptography. To bind the data owner's credentials to the public key of the Trust Point, we use the OAuth protocol [12]. The data owner takes the role of the OAuth *resource owner*, whereas the Trust Point is the *client* and the Cloud PaaS represents the *server* (see Figure 2). To trigger the binding process, the data owner sets up a secured connection with the Trust Point via TLS. The Trust Point then redirects her to the Cloud platform. However, the Cloud platform lacks an identity of the Trust Point that it can authorize for subsequent access, as the Trust Point is unknown to the Cloud platform at this time. Hence, the Trust Point encodes the fingerprint of its public key (i.e., its hash digest) in the redirect of the data owner to the Cloud platform. This enables the Cloud platform to store a mapping from the data owner's username to the public key fingerprint of the Trust Point, if the data owner authorizes the Trust Point at the Cloud platform. After the authorization procedure in the Cloud platform has been completed, the platform refers the data owner back to the Trust Point.

As a result of this second redirect, the Trust Point establishes a secure connection with the Cloud platform using its public key that corresponds to the previously exchanged fingerprint. The use of the mutual authentication variant of the TLS handshake in this connection establishment enables the Cloud platform to verify that the Trust Point has been correctly bound and authorized by the data owner before. Only then the Cloud platform accepts sensor data forwarded by this Trust Point.

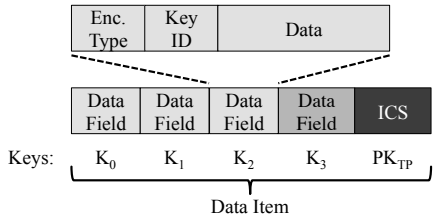


Figure 3. Object security consists of encryption of individual data fields (representing meta-data and raw data) and an integrity checksum (ICS) covering the complete data item. Each data field may be encoded using a key with a different ID and a different algorithm allowing for different confidentiality mechanisms to be used on individual data fields.

B. Object Security from Trust Point to Service

Transport security is terminated when sensor data is received in the Cloud. As a result, the transport security mechanisms are stripped from the sensor data and plain data would reside unprotected within the Cloud platform. This leaves the data stored in the Cloud open to several forms of attacks as discussed in Section II.

To achieve end-to-end security from the Trust Point to an authorized service and during storage, the Trust Point complements sensor data with additional object security mechanisms before transmitting them securely to the Cloud. In consequence, the plain information of sensor data can neither be accessed nor modified undetectably by an unauthorized third party. Furthermore, the employed integrity protection cryptographically ensures the accountability of sensor data to a specific data owner even after stripping the transport security mechanisms. Our approach to object security is comparable to Digital Rights Management (DRM) [13] when treating the Cloud service as end-user devices in the DRM case. However, the main difference to our solution is that we do not require enforcement of data access control on the service side.

Typically, a data item generated by a single sensor reading comprises of multiple data fields, i.e., raw measurement values and meta data such as location and time. These different fields can demand for different levels of protection. For example, the raw temperature readings of a private weather station may not require confidentiality protection while sensitive meta-data such as location information does. We design for this fact by supporting different protection schemes for the different data fields of a data item (see Figure 3). To realize fine-grained access control, individual protection keys can be used for different time spans (e.g., hourly, daily, and monthly).

In its simplest form, confidentiality protection is achieved with symmetric key encryption such as AES. To support search and sort operations on stored data, e.g., on time stamps, our object security design also allows the use of deterministic and order-preserving encryption [14], [15]. Likewise, it supports the use of efficient forms of homomorphic encryption for selected operations on encrypted data (e.g., sum or average) [14].

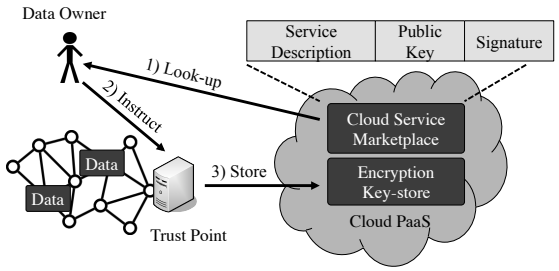


Figure 4. Service authorization by the data owner. The data owner looks up a service and its service description in the Cloud Service Marketplace. She then instructs her Trust Point to encrypt her data for the selected service. The Trust Point stores the encryption keys for this service in the encryption key-store.

Integrity protection in SensorCloud is based on asymmetric key cryptography. Specifically, the Trust Point signs each data item with its private key. Hence, while object encryption is performed on data fields, integrity protection spans the complete data item.

C. Service Assurance and Data Access Granting

The object security mechanisms employed in SensorCloud prevent access to the data owner's data items in the Cloud. Thus, she needs to authorize services if she wants them to process her data. To enable the data owner to perform an informed decision regarding which service to authorize, Cloud services provide a *service description* about themselves (e.g., in a Cloud service market-place) as depicted in Figure 4. This description contains high-level information about the kind of data it operates on and the purpose of the service. The conformance of the service to the service description must be verified by the Cloud provider or a trusted third party before approving the service to the Cloud platform similar to current practices on the smart phone market. If the data owner agrees with the service description, she needs to provide the encryption keys used for the protection of the data fields that she wants to share. This is done by instructing the Trust Point to encrypt the respective encryption keys with the public key of the service and to transmit this secured information to the *encryption key store* located in the Cloud. The required public key of the service is provided as a part of the service description. Authorized services request the encryption keys from the key store and decrypt them with their private keys. Similarly, new keys for all authorized services are pushed to the key store by the Trust Point whenever the encryption keys change.

D. Multi-layer Tenancy Separation

The transport and object-based encryption mechanisms ensure separation of sensor data from multiple tenants during transmission and storage. As the run-time context of a service contains sensitive information (e.g., decryption keys and the data owner's decrypted sensor data), encryption does not suffice to separate the tenant data during processing [16]. Thus, our platform isolates different tenants securely and by default in secure Service Execution Environments (SEE), as

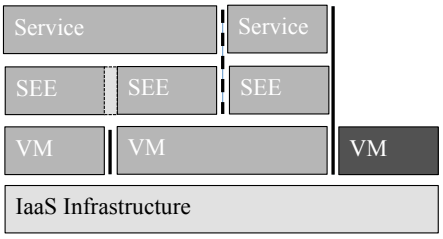


Figure 5. Multi-layer tenant separation by Secure Execution Environment (SEE) in the Cloud PaaS layer (dashed line) and by Virtual Machines (VMs) in the Cloud IaaS layer (solid line).

depicted in Figure 5. The SEEs are the only place where data items are unencrypted inside the Cloud platform. They ensure that services are not able to leave their SEE or interfere with services in other SEEs.

One requirement for our SEE design is that the platform guarantees tenant separation while scaling the number of services in a wide range. Typically, individual services for a few thousand data owners will be served from separate SEEs on the same physical machine. Likewise, the joined computing power of many physical machines will be required if services operate on huge amounts of data from various data owners.

Tenant separation relying purely on virtualization at the IaaS layer does not achieve this scalability. Due to memory restrictions, typical hypervisors can execute some tenth of virtual machines (VMs) on the same physical hardware. When the same application is run in parallel, aggressive memory sharing can be used to reduce the memory overhead of each VM. However, even then the total number of VMs on the same physical hardware is limited to a few hundreds [17]. Thus, implementing the SEE as a VM and having one VM per service is not feasible for a scenario with several thousand instances. Hence, the SEEs need to be implemented by our SensorCloud platform on top of an operating system running on instances (typically VMs) offered by the IaaS. We still rely on tenant separation by the computing part of the IaaS to isolate the SensorCloud PaaS provider from other IaaS users. Technically, this can be realized with a trusted hypervisor or simply by a process that only deploys VMs of the SensorCloud PaaS provider on the same physical hardware. The PaaS provider should ensure the adherence of the IaaS provider to this requirement, e.g., via regular audits required by legal agreements. As data in transit and storage is protected by the object security mechanisms, there are no specific separation requirements for the storage and networking subsystems of the IaaS layer.

Because implementing SEEs as VMs is not feasible in our scenario, we have to look for a more fine-grained approach. Given Java as a language for service development on the platform, one option would be to leave tenant separation to the Java VM (JVM) layer. However, neither typical JVMs nor Java EE application servers guarantee separation of executed applications. Security and isolation weaknesses

of JVMs were investigated, e.g., within the context of OSGi [18]. Recent research projects like I-JVM [19] are addressing these issues and are promising approaches for the future. However, we believe that currently the kernel and the operating system are the best means for enforcing tenant separation. Hence, our design is based on a multitude of JVMs running in parallel in isolated containers as different system processes for different tenants. The containers guarantee access to and oversee usage of system resources such as CPU, memory, and I/O. They can be implemented as BSD jails or Linux control groups.

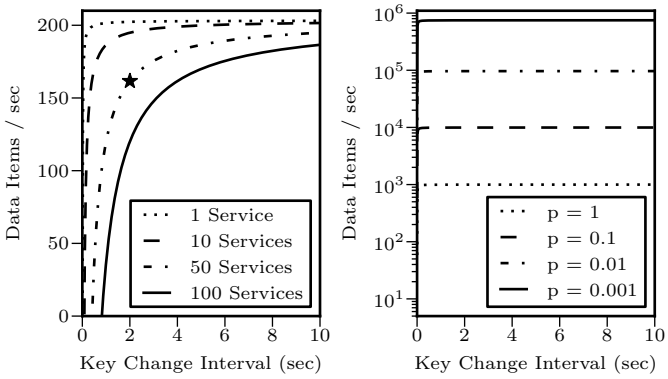
To enforce the usage of object security mechanisms by services, a dedicated object I/O API for services is defined in the SEE while forbidding any other network and storage I/O. Furthermore, the decryption of incoming and the encryption of outgoing data is provided as an (open source) library functionality that is used within the run-time context of the actual service. This architecture eases the development of secure services as it relieves the programmer from the burden of manual key and encryption handling.

IV. Evaluation

For the performance estimation of our design, we implemented a basic prototype that consists of a Trust Point and a Cloud platform. The Trust Point is a Linksys WRT160nl commodity router with 400 MHz that runs the Linux-based operating system OpenWRT. To allow for independent verification of our results, we use the Amazon EC2 IaaS platform with 64-bit instances of type large and Ubuntu 12.04 as the operating system for our Cloud measurements. Note that Amazon EC2 currently does not guarantee the desired level of user isolation. However, it can be achieved in our early IaaS prototype based on OpenStack by using corresponding policies for the filter scheduler.

Regarding the Trust Point-based architecture, SensorCloud involves performance trade-offs with respect to computations and storage. Furthermore, the multi-layer tenancy separation at the Service Execution Environment-level requires additional memory resources compared to pure virtualization on the IaaS layer. We now quantify and analyze these overheads.

Choice of cryptographic primitives: For our evaluation, we chose cryptographic primitives with at least 112 bit security providing data security up to the year 2030 according to NIST [20]. Furthermore, our cipher choice was guided by the aim to lower the computational burden on the Trust Point. The Trust Point constantly performs three types of cryptographic operations: i) per-data field encryption, ii) per-data item signing, and iii) per-authorized service encryption of data encryption keys. For the data field encryption, we use AES with 128 bit keys in CBC mode. Unless specified otherwise, we restrict our analysis to sensor data with a single data field, as the AES operations only represent a marginal overhead in our system. For signatures, we use



(a) Gateway performance (b) Cloud service performance

Figure 6. The left figure shows the data throughput of the *gateway* depending on the number of authorized services and the key change interval. In the right figure, the performance of a *Cloud service* depending on the integrity validation probability p and the key change interval is shown.

the ECDSA scheme with NIST curve P-224. Finally, the Trust Point encrypts data encryption keys with RSA using 2048 bit keys as RSA provides us with an advantageous performance asymmetry for public-key operations. However, future revisions of our architecture may replace RSA by elliptic curve encryption mechanisms such as ECIES.

Performance overhead: The performance overhead of the transport security mechanisms employed in SensorCloud have been well studied [21]. Hence, we focus our evaluation on the object security mechanisms of SensorCloud. Our object security mechanisms introduces a *computational overhead* on both the Trust Point and the Cloud platform. This overhead is introduced by the per-data field encryption and the per-data item integrity protection at the Trust Point. Likewise, services must decrypt encrypted data fields before processing them and may need to verify the integrity of stored data upon look-up. Moreover, the Trust Point changes the encryption keys periodically based on the desired granularity of the data access control. Accordingly, services need to decrypt the data encryption keys with their private key after each key change. The performance estimates of these operations are based on the OpenSSL implementation.

The results indicate that already at an access granularity of 2 seconds, our low-end Trust Point can process about 160 data items per second if 50 services are authorized to access these items (see star in Figure 6(a)). Only access granularities in the order of several hundred milliseconds (see Figure 6(a) with a key change interval close to 0 seconds) are infeasible. To bring these numbers into perspective, assume a per-sensor sampling rate of 1 measurement per second. In this case, the SensorCloud security architecture allows a low-end Trust Point to protect sensor data for over 150 nodes. If higher sampling rates or larger network sizes need to be supported, the hardware of the Trust Point can easily be scaled up or hardware support for the cryptographic algorithms could be added in order to achieve throughputs

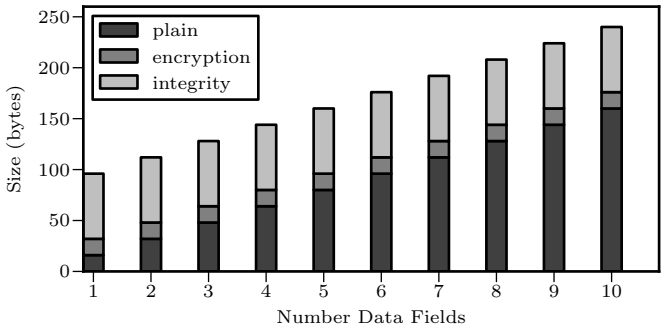


Figure 7. The storage overhead of the object security mechanism for each data item stays constant for an increasing number of data fields.

that are $1-2$ orders of magnitude higher. Specifically, the calculation of the data integrity mechanism denotes the limiting factor of the sensor data processing at the Trust Point. Modern multi-core CPUs allow to perform these calculations in parallel and at higher core speeds compared to a commodity router. Moreover, the number of data fields per data item only have a marginal influence on the maximum throughput of the Trust Point as symmetric encryption can be computed efficiently at 0.004 ms per data field. Hence, the SensorCloud security architecture also scales well with an increasing number of data fields.

A Cloud service may either access a live stream of a specific sensor or process sensor data that is stored in the Cloud (e.g., for services based on histories of sensor data). A service may likewise aggregate data from multiple sensors. Our results suggest that a service can process about 1000 data items per second if it verifies the integrity of all items (see Figure 6(b) for $p = 1$). This throughput suffices for small-scale services that operate on live data. However, large-scale services that aggregate data from multiple sensors require higher throughputs. Data throughput can be increased considerably, if the Cloud provider is trusted to a large extend. In this case, a Cloud service may only need to verify the data integrity for a random sample. For example, the data throughput increases to about 9900 or 96300 if the service randomly validates the integrity of $p = 0.1$ or $p = 0.01$ of the processed items respectively. In contrast, if deterministic verification is required for an untrusted Cloud storage provider, the data owner may use a dedicated third-party service that runs in the background and continuously verifies the data integrity. This allows for a continuous data integrity verification without the need to download the sensor data from the Cloud. Overall, the results show that the performance of our design scales as intended for our scenario.

Storage overhead: We distinguish two kinds of storage overhead in our evaluation. The first type of overhead is generated by the object security mechanisms themselves, i.e., sensor data encryption and integrity protection. The second type results from the encryption key storage in the Cloud. We restrict our estimation of the storage overhead to the pure overhead of the cryptographic ciphers and assume data fields that fit into a single 16 byte block. The specific storage overhead strongly depends on the exact encoding of the data fields and their respective information.

		Key Change Interval					
		1 second	1 minute	1 hour	1 day	1 week	1 month
Data Fields	1	4.33GB	0.07GB	1.23MB	0.05MB	7.50KB	1.75KB
	5	21.63GB	0.36GB	6.15MB	0.26MB	37.50KB	8.75KB
	10	43.26GB	0.72GB	12.30MB	0.51MB	75.00KB	17.50KB

Table I
KEY STORAGE OVERHEAD

The calculated storage overhead of our proposed object security mechanism is shown in Figure 7. The overhead consists of a 16 byte initialization vector per-data item for the data field encryption and a 64 byte integrity protection checksum. The encryption of the individual data fields does not add further overhead. Consequently, the cryptographic storage overhead of each data item is constant and amounts to 80 byte in addition to the actual sensor data. This overhead is well manageable with the elastic storage resources of today's Cloud offers.

The Cloud platform securely stores the encryption keys of protected data items in its dedicated key store for each service that is authorized to access a data item. Hence, the encryption key storage overhead in the Cloud strongly depends on the key granularity as well as the number of connected sensors and services that are authorized to access the data of the sensor. Tab. I depicts the calculated per-sensor storage overhead for different key granularities for one month worth of sensor data. Similar to the processing overhead of the employed object security mechanisms at the Trust Point, the key storage overhead approaches practical sizes with decreasing data access granularities, e.g., 70 MB at a per minute granularity for each data field. Note that old sensor data may be stale or may not have the same informational value for the data owner or her authorized services when compared to recently collected data. Hence, the data owner may delete or aggregate old data in order to decrease the total storage overhead in the Cloud with respect to sensor data, object security, and encryption key storage. We conclude that the storage overhead of SensorCloud is well manageable by the elastic storage capabilities provided by a Cloud platform.

SEE memory overhead: We developed an early SEE prototype for the evaluation of the memory overhead for the service isolation. An SEE consists of a dedicated OpenJDK JVM that is executed inside an application-level Linux Secure Container (LXC). LXC guarantees secure isolation of the processes (i.e., the JVMs) and allows to define quotas for processing, memory, and I/O usage. As we are interested in the memory overhead of the SEE itself, we used a minimal

test service written in Java for our evaluation. The memory consumed by one SEE instance and the test service together amounts to roughly 7 MB. Consequently, we were able to launch more than 1000 SEEs in parallel on one EC2 large instance with 7.5 GB of RAM. As IaaS environments like Amazon provide instances with up to about 70 GB of RAM, we can safely assume that we can run more than 10,000 SEEs with different services at the desired level of separation on a single physical machine.

V. Related Work

For our discussion of related work, we distinguish the following three research directions: i) architectures involving a trusted third-party, ii) secure operations on outsourced data, and iii) other related approaches to secure Cloud computing.

Architectures utilizing a trusted third-party similar to our Trust Point have been proposed in a wide range of scenarios. In the context of Wi-Fi-sharing communities, a trusted relay can be used to mitigate security, privacy, and legal issues [22]. However, such data flow-focused approaches typically restrict themselves to the use of transport security and do not consider the object security that is necessary in our scenario. To guarantee privacy in intelligent energy networks in Germany, a trusted gateway has been specified [23]. Although our security architecture shows similarities to this approach, its PKI-based approach does not allow for similarly fine-granular access control on sensor data.

A number of architectures involving a trusted third-party have been proposed in the context of Cloud computing. Kamara et al. [24] propose an architecture that resembles ours with respect to fact that a trusted gateway encrypts outbound data and manages access policies. However, the authors do not consider the secure computation of data by Cloud services and require the data consumer to request access tokens from the gateway. Hence, their approach does not consider the necessary Cloud service isolation. Furthermore, data stored in the Cloud becomes unavailable if the gateway is offline. The Twin Clouds architecture [25] uses Garbled Circuits to encrypt both outsourced data and programs in a trusted (private) Cloud. After this demanding per-data item setup phase, computations can be performed on an untrusted public Cloud platform. However, encrypted programs only perform simple operations and need to be re-encrypted after each execution. Similar to our security architecture, Pearson et al. [26] introduce a Cloud design that focuses on fine-grained access control for outsourced data by the data owner. However, while their approach focuses on sticky policies that are enforced by an external trust authority, our contribution includes the introduction of the Trust Point and its binding with the Cloud, a flexible design for object security on sensor data, and the incorporation of isolation mechanisms in the Cloud at the service-level.

Secure operations on outsourced data can further be classified into secure data querying and secure computations. The field of secure data querying focuses on retrieving encrypted data in a structured way in order to allow, e.g., range queries or keyword searches [14], [24], [15]. In contrast, secure computations allow for direct computations on encrypted data. A prominent example of the latter cryptographic primitive is (fully) homomorphic encryption [6], [14]. However, especially fully homomorphic encryption still is highly inefficient regarding the computational overhead and key sizes [7]. As described in Section III-B, our flexible design of data object security mechanisms allows to incorporate efficient approaches for querying and processing encrypted data.

A number of *other related approaches to secure Cloud computing* have been proposed in the past. They either build on the fundamental idea of using trusted hardware components in Cloud environments [27], [4] or of performing data processing locally. However, TPM-based approaches require hardware support by the IaaS layer, whereas the IaaS requirements of our architecture can be implemented at the management level. Furthermore, the binding of trusted components to specific hardware makes the migration of virtual instances across multiple hardware platforms challenging [5]. When performing data processing locally, the Cloud is merely used as a storage device [24], [11] and computations are done outside the Cloud within the trusted domain of the data owner [7]. In contrast to our approach, these solutions do not take advantage from the elastic computational resources that are offered by the Cloud.

VI. Conclusion

When outsourcing storage and processing of sensor data to the Cloud, multiple possibly unknown or untrusted stakeholders become involved. In this paper, we identified the specific threats that arise from this uncertain involvement. Our proposed SensorCloud security architecture counters these threats by allowing the data owner to stay in control over her data even in a Cloud scenario. To this end, we introduce a Trust Point as a new logical entity that is located at the border of the sensor network and acts as a bridge between the security domain of the sensor network and the Cloud. The Trust Point i) implements transport security mechanisms for communication with the Cloud, ii) applies object security mechanisms to outbound data items, and iii) performs key management for authorized services. To mitigate leakage of sensitive information from the run-time contexts of services, we additionally propose the use of isolation mechanisms all the way up to the service-level. Our evaluation shows that our proposed SensorCloud security architecture has an adequate performance for the intended scenario and that the involved storage and memory overheads can be handled effectively. This makes the user-controlled storage and processing of sensor data a promising extension of today's Cloud offers.

ACKNOWLEDGMENT

The authors would like to thank the members of the SensorCloud consortium for the inspiring discussions on the concepts described in this paper. The SensorCloud project is funded by the german Federal Ministry of Economics and Technology under the project funding reference number 01MD11049. The responsibility for the content of this publication lies with the authors.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Comm. Mag.*, vol. 40, no. 8, 2002.

[2] S. Pearson and A. Benameur, "Privacy, Security and Trust Issues Arising from Cloud Computing," in *Proc. IEEE CloudCom*, 2010.

[3] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling Data in the Cloud: Outsourcing Computation without Outsourcing Control," in *Proc. ACM CCSW*, 2009.

[4] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards Trusted Cloud Computing," in *Proc. USENIX HotCloud*, 2009.

[5] D. Wallom, M. Turilli, G. Taylor, N. Hargreaves, A. Martin, A. Raun, and A. McMoran, "myTrustedCloud: Trusted Cloud Infrastructure for Security-critical Computation and Data Managment," in *Proc. IEEE CloudCom*, 2011.

[6] C. Gentry, "Computing Arbitrary Functions of Encrypted Data," *Commun. ACM*, vol. 53, no. 3, 2010.

[7] G. Danezis and B. Livshits, "Towards Ensuring Client-Side Computational Integrity," in *Proc. ACM CCSW*, 2011.

[8] G. Stoneburner, "Underlying Technical Models for Information Technology Security," NIST Special Publication 800-33, National Institute of Standards and Technology, 2001.

[9] M. Jensen, J. Schwenk, N. Gruschka, and L. Iacono, "On Technical Security Issues in Cloud Computing," in *Proc. IEEE CLOUD*, 2009.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," in *Proc. CCS*, 2009.

[11] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," in *Proc. ACM CCS*, 2009.

[12] E. Hammer-Lahav, "The OAuth 1.0 Protocol," RFC 5849 (Informational), IETF, 2010.

[13] E. Becker, W. Buhse, D. Günnewig, and N. Rump, Eds., *Digital Rights Management*. Springer, 2003.

[14] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," in *Proc. ACM SOSP*, 2011.

[15] D. Boneh and B. Waters, "Conjunctive, Subset, and Range Queries on Encrypted Data," in *Proc. TCC*, 2007, LNCS 4392.

[16] M. Van Dijk and A. Juels, "On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing," in *Proc. USENIX HotSec*, 2010.

[17] L. Martignoni, P. Poosankam, M. Zaharia, J. Han, S. McCamant, D. Song, V. Paxson, A. Perrig, S. Shenker, and I. Stoica, "Cloud Terminal: Secure Access to Sensitive Applications from Untrusted Systems," in *Proc. USENIX ATC*, 2012.

[18] P. Parrend and S. Frenot, "Security benchmarks of OSGi platforms: toward Hardened OSGi," *Softw: Pract. Exper*, vol. 39, 2009.

[19] N. Geoffray, G. Thomas, G. Muller, P. Parrend, S. Frénot, and B. Folliot, "I-JVM: a Java Virtual Machine for Component Isolation in OSGi," in *Proc. IEEE/IFIP DSN*, 2009.

[20] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for Key Management – Part 1: General (Revision 3)," NIST Special Publication 800-57, National Institute of Standards and Technology, 2012.

[21] C. Coarfa, P. Druschel, and D. S. Wallach, "Performance Analysis of TLS Web servers," *ACM Trans. Comput. Syst.*, vol. 24, no. 1, 2006.

[22] T. Heer, T. Jansen, R. Hummen, S. Götz, H. Wirtz, E. Weingärtner, and K. Wehrle, "PiSA-SA: Municipal Wi-Fi Based on Wi-Fi Sharing," in *Proc. ICCCN*, 2010.

[23] "Protection Profile for the Gateway of a Smart Metering System," v01.01.01(final draft), Federal Office for Information Security, Germany, 2011.

[24] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," in *Proc. RLCPS*, 2010, LNCS 6054.

[25] S. Bugiel, S. Nürnberger, A.-R. Sadeghi, and T. Schneider, "Twin Clouds: Secure Cloud Computing with Low Latency," in *Proc. IFIP CMS*, 2011, LNCS 7025.

[26] S. Pearson, M. Mont, L. Chen, and A. Reed, "End-to-End Policy-Based Encryption and Management of Data in the Cloud," in *Proc. IEEE CloudCom*, 2011.

[27] W. Itani, A. Kayssi, and A. Chehab, "Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures," in *Proc. IEEE DASC*, 2009.