International Journal of Grid and High Performance Computing

October-December 2013, Vol. 5, No. 4

Table of Contents

Special Issue on Cloud Computing Technology and Science

Research Articles

The *International Journal of Grid and High Performance Computing* is indexed or listed in the following: ACM Digital Library, Bacon's Media Directory, Cabell's Directories, Compendex (Elsevier Engineering Index), DBLP, GetCited, Google Scholar, INSPEC, JournalTOCs, MediaFinder, SCOPUS, The Standard Periodical Directory, Ulrich's Periodicals Directory

Maintaining User Control While Storing and Processing Sensor Data in the Cloud

Martin Henze, Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany

René Hummen, Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany

Roman Matzutt, Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany

Daniel Catrein, QSC AG, Cologne, Germany

Klaus Wehrle, Communication and Distributed Systems, RWTH Aachen University, Aachen, Germany

ABSTRACT

Clouds provide a platform for efficiently and flexibly aggregating, storing, and processing large amounts of data. Eventually, sensor networks will automatically collect such data. A particular challenge regarding sensor data in Clouds is the inherent sensitive nature of sensed information. For current Cloud platforms, the data owner loses control over her sensor data once it enters the Cloud. This imposes a major adoption barrier for bridging Cloud computing and sensor networks, which we address henceforth. After analyzing threats to sensor data in Clouds, the authors propose a Cloud architecture that enables end-to-end control over sensitive sensor data by the data owner. The authors introduce a well-defined entry point from the sensor network into the Cloud, which enforces end-to-end data protection, applies encryption and integrity protection, and grants data access. Additionally, the authors enforce strict isolation of services. The authors show the feasibility and scalability of their Cloud architecture using a prototype and measurements.

Keywords: Access Control, Architecture, Cloud, End-to-End Protection, Platform, Security, Sensor Network

INTRODUCTION

Continuous advances in the areas of ubiquitous computing and sensor networks (Akyildiz et al., 2002) blur more and more the boundaries between the physical and the digital world. At the same time, Cloud computing turned into an established paradigm for outsourcing data storage and computation resources. These two trends, ubiquitous sensing and Cloud computing, complement each other naturally. Sensor networks are an ideal solution to collect information about the physical environment, but they typically lack the resources to store and

DOI: 10.4018/ijghpc.2013100107

process collected data over long periods of time. However, Cloud computing provides nearly unlimited storage and computing resources elastically. Additionally, Cloud-based services could make use of the data from a large number of sensor networks. How these services could be used is illustrated by the following example: Assume private weather stations that do not only provide a local view on current sensor readings, but also transmit their measurements to a forecast service running in the Cloud. The Cloud service now is able to process and aggregate the received data and use it in a Cloud-based weather simulation in order to generate an accurate forecast for a specific region.

However, one major concern with this approach is inherent to sensor data: it often contains private or otherwise sensitive (meta-) information. The weather forecast service may, for example, require the location in addition to the raw, insensitive temperature values. While the data owner may be willing to share this sensitive information with selected services, she often does not trust the Cloud provider or other services. Thus, she may refrain from using Cloud-services that are based on her sensor data. As previously identified for related scenarios (Chow et al., 2009; Henze et al., in press; Pearson & Benameur, 2010), these adoption barriers arise due to the loss of control of the data owner over her data. A number of approaches aiming at providing secure data storage and computation in the Cloud have been proposed. Typically, these approaches focus on providing hard security guarantees by using technologies such as trusted platform modules or homomorphic encryption (Gentry, 2010; Santos et al, 2009; Wallom et al., 2011). These approaches are, however, inadequate for storing and processing sensitive sensor data in the Cloud. Either they do not provide the necessary user control over outsourced data in the Cloud or they introduce excessive encryption overhead, especially when applied to comparably small sensor data (Danezis & Livshits, 2011). Hence, we see the need for a practically viable approach for storing and processing sensor data in the Cloud.

This work is structured as follows: After this introduction, we provide a detailed overview of related work with a discussion how our proposals differentiate. Afterwards, we present threats originating from various entities that arise when outsourcing storage and processing of sensitive sensor data to the Cloud. In order to address these threats, we propose a security architecture for user-controlled storage and processing of sensor data in the cloud. Our proposed security architecture provides a Platform-as-a-Service (PaaS) for the execution of services that operate on sensor data in accordance with the necessary security requirements. For this purpose we offer i) early protection of sensor data starting in the sensor network, ii) user-controlled access granting for selected services, and iii) strict service isolation within the Cloud platform. With these measures we enable the data owner to stay in control over who may access her data and, thus, make outsourcing sensor data to the Cloud viable.

RELATED WORK

We structure our discussion of related work into 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.

In a wide range of scenarios, architectures utilizing a trusted third-party similar to our Trust Point architecture have been proposed. However, these approaches typically restrict themselves to securing the transport of data and do not consider the object security that is crucial for our scenario. The Federal Office for Information Security in Germany (2011) specified a trusted gateway in order to guarantee privacy in intelligent energy networks. Our security architecture shows some similarities to this approach. However, our architecture allows a much more fine-granular access control for data. There are also a number of architectures involving a trusted third-party that have been proposed in the context of Cloud comput-

ing. Kamara and Lauter (2010) propose an architecture similar to ours with respect to a trusted gateway encrypting outbound data and managing access policies. However, they do not consider the secure processing of data in the Cloud. Additionally, they require requesting access tokens from the gateway in order to access data. Hence, in contrast to our approach, data stored in the Cloud is only available when the gateway is reachable. The Twin Clouds architecture by Bugiel et al. (2011) utilizes Garbled Circuits for encrypting both data and programs in a trusted environment before passing them to the untrusted public Cloud. After this demanding setup phase, which has to be performed per data item, computations can be executed in the untrusted Cloud. However, the encrypted programs are limited to simple operations and require re-encryption after each execution. Pearson et al. (2011) introduce a Cloud design similar to ours that focuses on fine-grained access control for outsourced data. While their approach focuses on sticky policies that have to be enforced by a trusted third-party, our solution introduces the Trust Point and its binding with the Cloud, suggests a flexible design for object security on sensor data, and incorporates isolation mechanisms at the service-level in the Cloud.

Secure operations on outsourced data can be distinguished into secure data indexing and secure computations. The area of secure data indexing focuses on accessing encrypted data in a structured way in order to allow, e.g., range queries or keyword searches (Boneh & Waters, 2007; Kamara & Lauter, 2011; Popa et al., 2011). In contrast, the field of secure computations deals with computations directly on encrypted data. One prominent example for this is (fully) homomorphic encryption (Gentry, 2010; Popa et al., 2011). However, especially when considering fully homomorphic encryption, high inefficiencies regarding computational overhead and key sizes can be observed (Danezis & Livshits, 2011). As we consider indexing and processing of encrypted data promising, our flexible design of data object security mechanisms allows incorporating these approaches.

There is also a number of other related approaches to secure Cloud computing. These either build on the idea of using trusted hardware components in Cloud environments (Itani et al., 2009; Santos et al., 2009) or of performing data processing outside the Cloud. Approaches utililizing trusted hardware components require extensive support by the IaaS layer, whereas the requirements of our architecture to the IaaS can be implemented at the management level. Additionally, binding trusted components to specific hardware instances makes migrating virtual instances across multiple hardware platforms a challenging task (Wallom et al., 2011). Performing data processing locally reduces the Cloud to a storage device (Bowers et al., 2009; Kamara & Lauter, 2010). In contrast to our approach, these solutions cannot profit from the elastic computational resources that the Cloud offers.

SCENARIO AND SECURITY CONCERNS

Sensor networks are typically dedicated and isolated networks, which collect information about their environment. A sink node that may either pre-process the raw data on-site or forward it directly to its actual consumer commonly collects the sensed raw data. 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 take become ambiguous as multiple entities contribute to the overall service provisioning. More precisely, data sent by the sensor gateway through the Internet towards the Cloud traverses the network backbone infrastructure as well as the IaaS and PaaS layers before being processed by a service (see Figure 1). 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 architecture must prevent access to information by the Cloud provider and third parties, unless the data owner explicitly agrees to share it.

An extensive study of the high-level threats for outsourced data in the Cloud and the building blocks of a trusted Cloud has already been performed (Hummen et al., 2012). This study identified data confidentiality and integrity, data accountability, service availability, and assurance as the main threats, which have to be tackled using legal agreements, processes, and technology. In this work, we focus our discussion to specific threats when outsourcing sensor data to the Cloud. To this end, we first introduce the entities involved in the storage and processing of sensor data in the Cloud and present assumptions regarding these entities. We then identify specific threats that originate from these entities. Based on these threats, we derive security goals our architecture has to fulfill.

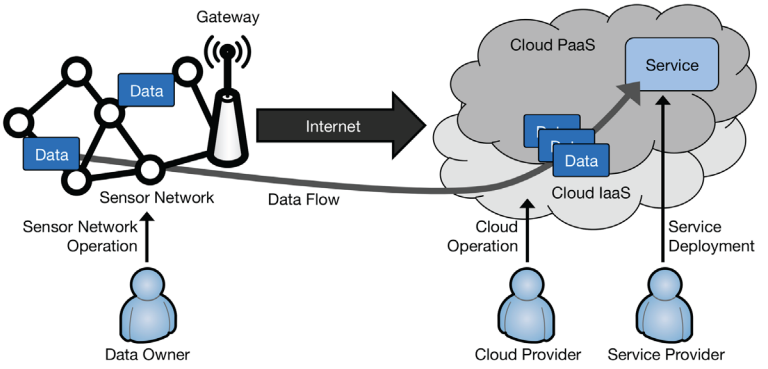
Entities

Sensor data that is owned by the data owner and outsourced to the Cloud is potentially exposed to different entities (see Figure 1). The *cloud provider* operates the Cloud and has the possibility to monitor any aspect of the Cloud at any time. Additionally, any *service provider*, which operates its service on top of the platform offered by the cloud provider, may try to access any sensor data via the service. Finally, all sensor data generated by the sensor networks are transmitted to the Cloud via the Internet. The same holds for communication between the Cloud and end users accessing the data. Hence, an *external attacker* might eavesdrop on connections between the Cloud and any other entity, with no other purpose but to obtain information or manipulate data. Before discussing threats regarding these entities in more detail, we first present the high-level assumptions and adversary models for our scenario and entities.

Assumptions and Adversary Models

We assume that the Cloud provider obeys to an adversary model similar to that of an *honest-but-curious* adversary (Kissner & Song, 2005). That means, it will operate technology, services, and interfaces as contractually agreed and will not actively spy into running services. However, it might try to learn as much as possible about the processed information and it might not guarantee long-term confidentiality of stored information. The service providers, on the contrary, are generally considered less trustworthy. This is due to the fact that the data owner cannot control which services are offered by the Cloud and those services may actively try to gain unconstrained access to sensor data that is not meant for disclosure to them. Some

Figure 1. Different entities are involved in the general scenario of outsourcing storage and processing of sensor data to the Cloud



level of trust in services can be gained after an audit of the service through the Cloud provider or a trusted third party. This is similar to the approach taken by today's app stores. Outside entities must be seen as malicious adversaries (Kissner & Song, 2005) that may perform arbitrary actions in order to break into communication flows.

Threats

We now discuss potential threats for sensor data in the Cloud, originating from the different entities with their adversary models. First, we investigate the cloud provider's possibilities to obtain sensor data without approval. Secondly, we have a look at attacks a service provider could launch. Finally, we discuss attacks that an external attacker could launch.

Threats from Cloud provider: In the following we differentiate three ways in which the honest-but-curious Cloud provider could try to access sensor data. Most intuitively, the Cloud provider can inspect the cloud storage that is used to persistently store sensor data in order to spy on the data. The Cloud provider is able to monitor any aspect of the Cloud at any time. Hence, it has the ability to inspect the Cloud's storage at any time. Another possibility for the Cloud provider to get unintentional access to sensitive data is by eavesdropping, e.g., when collecting data (e.g., network statistics) on internal communication to satisfy its curiosity. This holds not only for the protected data but also for other sensitive information, i.e., information needed to decrypt protected data and computation results. Finally, the Cloud provider could (unintentionally) spy on service information that leaked out of a run-time context or that survived the end of the run-time context of a service. This may be data left in memory or in temporary files on disks after a service released the respective resources. Note that the honest-but-curious adversary assumed here does explicitly not monitor the private runtime memory of a service in order to obtain any of the information mentioned above. We also assume that the access to physical resources of the Cloud or virtual resources allocated by services is sufficiently restricted to make the direct investigation of those resources infeasible for others than the Cloud provider (Lombardi & Di Pietro, 2011).

Threats from service provider: We assume that service providers potentially are more aggressive attackers than the cloud provider. Therefore, we explicitly consider malicious service providers that actively try to access sensor data that they are not authorized to process. There are essentially two threats originating from malicious services: access escalation and service identification spoofing. We first discuss access escalation. To protect the Cloud, the Cloud provider has to manage the permissions of its customers in the Cloud. Services can attempt to bypass their access restrictions in various ways. A service could try to obtain additional permissions by exploiting errors in the implementation of the Cloud's permission management. Furthermore, a service could launch side-channel attacks as described by Ristenpart et al. (2009) and Zhang et al. (2012) in order to obtain information about other services that are instanced within another virtual machine on the same physical device in the Cloud. Additionally, the service can try to break out of its virtualization environment entirely, gaining access to the physical device the service is instanced on. Using the second threat, service identification spoofing, a service impersonates another service and thus obtains unauthorized access to sensor data.

Threats from an external attacker: In our scenario, the Cloud is publicly available via the Internet. Hence, all entities in our scenario are subject to the commonly known security threats the Internet has yielded over time. In this section, we describe attacks a malicious external attacker could launch against the Cloud infrastructure used to realize the service. These are eavesdropping on connections, impersonation of the Cloud, as well as data and service forgery. At first, we discuss eavesdropping on connections to the Cloud. Whenever data are communicated between a sensor network or a user and the Cloud, messages have to be

sent via the Internet. An external attacker can eavesdrop on the established communication channel in order to obtain confidential information. The next threat arises from the external attacker trying to impersonate the Cloud. This way he would receive all the information any entity sends to the Cloud. For data forgery, a malicious attacker could try to manipulate the sensor data outsourced to the Cloud. This is not only limited to the sensor data being stored but also applies to the results of a service which operates on sensor data. A similar threat is service forgery. An external attacker can appear as man-in-the-middle when a service provider is about to deploy its service to the Cloud. His malicious service, which then again could attempt any attack described earlier, would more likely be accepted and authorized by data owner than a genuinely deployed malicious service.

Security Goals

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. Our main security goal is the control of the data owner over her sensor data when outsourcing storage and processing to the Cloud. To enable this control, our security architecture primarily has to meet the following high-level requirements.

- **Authentication:** All entities must be able to verify the identity of all other entities it interacts with.
- **Secure Transport and Storage:** All data must be secured properly during transport and storage.
- **Data Confidentiality and Integrity:** The data owner must be able to control who can access her data in- and outside the Cloud platform. Additionally, the data owner and Cloud services must be able to verify that data has not been manipulated.
- **Trusted Services and Service Execution:** The data owner must be sure which service operates on her data and what the service does with her data.

Based on these security goals, we continue with the presentation of our security architecture.

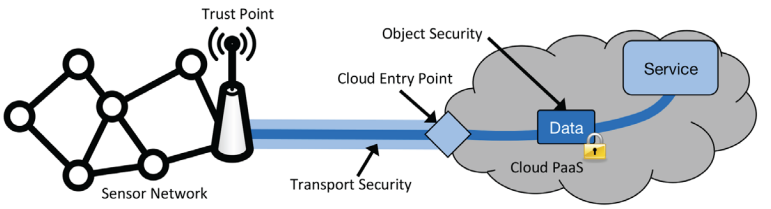
SECURITY ARCHITECTURE

In order to address the presented threats and meet the stated security goals, we developed a security architecture for the user-controlled storage and processing of sensor data in the Cloud. As stated earlier, the main goal of our work is to establish control of the data owner over her sensor data when outsourcing storage and processing to the Cloud. To achieve this goal, we identified four components, which we will discuss in the following. Before we dive into details, we first give a high-level overview of these components. We start our presentation with the bridging of the sensor network and the Cloud. This includes securing the transport between the sensor network gateway and the Cloud entry point. As transport security is terminated as soon as the Cloud receives the sensor data, we additionally add object security. This allows end-to-end security from the sensor network to an authorized Cloud service as well as during storage. Figure 2 illustrates the different protection scopes of transport and object security. In order to grant authorized services access to data, the data owner has to provide the keys used for the object security to the service. We discuss possible methods for this key management task in a Cloud setting. Finally, multi-layer tenancy separation allows protecting data not only during transport and storage, but also during processing.

Bridging the Sensor Network and Cloud Domains

All data leave the trusted sensor work at a common point of control: the gateway that connects the sensor network with the outside world (e.g., the Internet). Hence, we realize our control and security mechanisms at this border point of the

Figure 2. Transport security is terminated as soon as data reaches the Cloud entry point. To protect the path between Cloud entry point and service, we additionally employ object security.



sensor network. Specifically, we introduce the Trust Point, a new logical entity on the gateway. It acts as a bridge between the trust domain of the sensor network and the Cloud and performs the following three tasks. First, on behalf of the data owner, it forwards the sensor data to the Cloud platform. Second, during the transmission to the Cloud, it protects the confidentiality and integrity of the forwarded data. 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 Cloud platform provider offers storage and processing resources to the data owner. To enable accountability (and billing), we require that data owners have to register with the Cloud platform. In order to allow the Trust Point to store data in the Cloud on behalf of the data owner, we have to bind its identity to the data owner's account in the Cloud. We refrain from using the data owner's credentials (e.g., username and password) on the Trust Point for security reasons. Instead, we identify the Trust Point using public key cryptography. We use the OAuth protocol (Hammer-Lahav, 2010) to bind the data owner's credentials to the public key of the Trust Point. In the OAuth model, the data owner takes the role of the OAuth resource owner, whereas the Trust Point is the OAuth client and the Cloud PaaS represents the OAuth server (see Figure 3). To perform the binding process, the data owner first connects to the Trust Point using a TLS secured connection. The Trust Point then redirects her to the Cloud platform. So far, the Trust Point is unknown to the Cloud platform and there is no identity provided which could be authorized for future access. In order to provide such an identity, the Trust Point adds the fingerprint of its public key (i.e., its hash digest) to the redirect request. After the data owner authorized the Trust Point at the Cloud platform, the Cloud platform is able to create a mapping from the data owner's username to the public key fingerprint of the Trust Point.

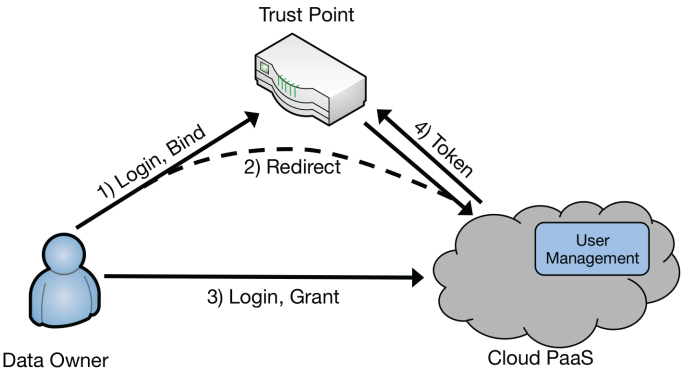
As soon as the authorization process has been completed, the platform redirects the data owner back to the Trust Point. Triggered by this second redirect, the Trust Point is now able to establish a secure connection with the Cloud platform. In order to identify itself, it uses its public key for the mutual authentication variant of the TLS handshake. As the Cloud now knows the Trust Point's public key, correct binding and authorization of the Trust Point can now be verified. Now, the Cloud platform can safely accept sensor data transmitted by this Trust Point.

Object Security from Trust Point to Service

As motivated earlier, transport security is terminated when the Cloud receives the sensor data (see Figure 2). At this point, the transport security mechanisms are stripped from the sensor data. Without further protection plain data would reside unprotected within the Cloud platform, which leads to several threats as discussed earlier.

To achieve end-to-end security from the Trust Point to an authorized service and during storage, the Trust Point adds additional object

Figure 3. In order to set up a secure and mutually authenticated connection between the Trust Point and the Cloud, the data owner triggers the OAuth-based binding procedure



security mechanisms to the sensor data before transmitting them securely to the Cloud. The plain information of sensor data can now neither be accessed nor modified undetectably by an unauthorized third party. Furthermore, the additional integrity protection cryptographically guarantees the accountability of sensor data to a specific data owner; even after the transport security has been terminated. This approach to object security is comparable to Digital Rights Management (DRM) (Becker et al., 2003) when treating the Cloud services 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.

A data item generated by a single sensor reading typically consists of multiple data fields, i.e., raw measurement values and meta data such as location and time. The type of data fields and meta data depends on the type of the sensor. A sensor attached to a windmill will most likely output data that is structured completely different compared to those of a sensor measuring vital signs in a hospital. In order to cope with the variety of different sensor data types, we propose to use JavaScript Object Notation (JSON) (Crockford, 2006) for representing and serializing sensor data. We intentionally do not restrict the format of these JSON objects. However, we assume the existence of certain fields, which are necessary for indexing the data, such as identifiers for sensor node and gateway, timestamp, and sensor data type. These different data fields (raw measurement values and meta data) can demand for different levels of protection. For example, the raw temperature readings of a private outdoor weather station may not require confidentiality protection while sensitive meta-data such as location information does. In order to address this fact, we support different protection schemes for the different data fields of a data item. We note that both, Trust Point and Cloud service need a formal description of a sensor data type (identified by the data type field). In the context of JSON, this can be achieved using JSON Schema (Galiegue et al., 2013). We propose to enrich these schema definitions by instructions on how a specific data field should be encrypted. For each data field it is thus possible to specify if and with which encryption algorithm this field has to be encrypted. Utilizing JSON has the nice benefit that the communication of encryption parameters such as an identifier for used encryption key or the initialization vector can easily be represented using JSON Web Encryption (JWE) (Jones et al., 2013). We thus instruct the Trust Point to replace the plain value with a JWE object containing the cipher text and encryption parameters. To further realize fine-grained access control, individual protection keys can be used for different time spans (e.g., hours, days, weeks).

The straightforward solution for confidentiality protection is symmetric key encryption, e.g., AES. Our security design also enables the use of order-preserving and deterministic encryption (Boneh & Waters, 2007; Popa et al., 2011) in order to allow search and sort operations on stored (encrypted) data. Similarly, efficient methods for homomorphic encryption (Popa et al., 2011) for selected operations on encrypted data (e.g., sum or average) can be used.

The integrity protection of our architecture is based on asymmetric key cryptography. To guarantee integrity of each data item, the Trust Point signs it with a private key. Thus, the integrity protection covers the complete data item.

Service Assurance and Data Access Granting

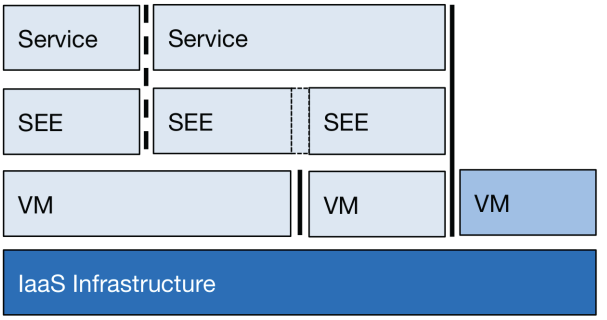
To access data items and individual data fields, services need to have the keys used to protect them. This access has to be authorized by the data owner. Data owners have to be empowered to perform an informed decision regarding the services to authorize. Hence, Cloud services come with a service description (e.g., in a Cloud service market-place). This description contains high-level information about the purpose of the service and how the service uses the data provided. Conformance of the service implementation to the service description must be assured, e.g., via an audit by the Cloud provider or a trusted third party, similarly to practices applied in today's app stores. The conformance is expressed via a cryptographic signature issued by the auditor and signing the service description and the service's public key. A data owner agreeing with the service description provides the encryption keys used for the protection of the data fields to the service after verifying the signature. This is achieved 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 key store's purpose is twofold. It offloads the Trust Point from the burden of frequent and repeated key requests causing expensive public key operations or the need to store a large number of keys. Additionally, it relaxes the requirement that the Trust Point needs to be continuously online. Connectivity to the Trust Point is only necessary to grant access to new tuples of key and service. Authorized services retrieve the encryption keys from the key store and decrypt them using their private keys. Similarly, for all authorized services, the Trust Point pushes new keys to the key store whenever the encryption keys change.

Multi-Layer Tenancy Separation

From the Trust Point on, tenant data is secured and securely separated during transport and storage by transport and object encryption. However, when processing data in services, sensitive information (e.g., decryption keys and the data owner's decrypted sensor data) has to be stored and processed in an unencrypted manner in the run-time context of the service (van Dijk & Juels, 2010). Thus, our platform introduces secure Service Execution Environments (SEE) as containers for service execution, as depicted in Figure 4. The SEEs are the only place where data items are unencrypted inside the Cloud platform. SEEs isolate different tenants from each other and services are not able to leave their SEE or interfere with services in other SEEs. The SEE design has to guarantee tenant separation while scaling the number of services in a wide range. Either, individual services for a few thousand tenants, i.e., data owners, may be served from separate SEEs on the same physical machine. Or, the joint computing power of many physical machines may be required for services operating on huge amounts of data from various data owners.

Tenant separation relying on virtualization at the IaaS layer does not achieve this scalability. Even when using aggressive memory sharing techniques, the total number of VMs on the same physical hardware is limited by memory restrictions to a few hundreds (Martignoni et al., 2012). Hence, using a separate VM as SEE for each tenant and service does not support scenarios with several thousand instances.

Figure 4. Multi-layer tenant separation is achieved using Virtual Machines (VMs) at the IaaS layer (solid lines) and Secure Execution Environments (SEEs) at the PaaS layer (dashed line)



Consequently, we implement the SEEs on top of an operating system, which is executed on Instances (typically VMs) offered by an IaaS provider. Note that tenant separation by the computing part of the IaaS is still required to isolate the PaaS provider from other IaaS users. This can be realized by a trusted hypervisor or simply by a process that ensures that only VMs of the PaaS provider are executed on the same physical hardware. Adherence of the IaaS provider to this requirement needs to be enforced, e.g., via legal agreements and regular audits. As data in transit and storage is protected by the object security mechanism, there are no specific separation requirements for the storage and networking subsystems at the IaaS layer.

When restricting to Java as a language for service development on the platform (which is not really desirable), an option would be to use the Java VM (JVM) as SEE. However, neither typical JVMs nor Java EE application servers guarantee proper separation of executed applications and respective tenants. Security and isolation weaknesses of JVMs were investigated, e.g., within the context of OSGi (Parrend & Frenot, 2009) and research projects like I-JVM (Geoffray et al., 2009) are addressing them. Even though, these might be promising approaches for the future, we believe that currently SEE implementation and tenant separation closer to the kernel and the operating system is favorable. Hence, we propose here to use OS level containers such as BSD jails or Linux control groups as SEEs. The containers guarantee access isolation to and oversee usage of system resources such as CPU, memory, and I/O. Another option for further investigation and research is to implement sandboxes via LLVM, e.g., by extending the approach on software fault isolation described by Sehr et al. (2010).

In any case, existing containers need to be extended to enforce the usage of object security mechanisms by services. A dedicated object I/O API for services is defined "at the boundary" of the SEE while any other network and storage I/O can be forbidden. Incoming and outgoing objects are automatically decrypted or encrypted by this API, respectively. Cryptographic operations and corresponding key handling happen automatically in the run-time context of the actual service and tenant. This architecture enforces object encryption and supports the development of secure services as it relieves the programmer from the burden of manual key and encryption handling.

EVALUATION

We implemented a basic prototype, consisting of a Trust Point, Cloud platform, and Cloud service, in order to estimate the performance and prove the feasibility of our design. We us a Raspberry Pi Model B with 256 MiB of RAM, a clock speed of 700 MHz, and Raspbian (a Debian-based Linux distribution for the Raspberry Pi)

as operating system as hardware platform for the prototype of our Trust Point. In order to allow others to verify our experimentation results, we use Amazon Web Services (2013) 1st generation EC2 64-bit instances of type large (M1.large) and Ubuntu 12.04 as the operating system for our performance measurements in the Cloud. We are aware that Amazon EC2 does currently not guarantee the level of user isolation we desire for our architecture. Thus, we have a second prototype running on OpenStack where the desired level of isolation can be achieved by configuring respective policies for the filter scheduler. Our Trust Point-based architecture involves performance trade-offs with respect to computation and storage resources. Additionally, the multi-layer tenancy separation at the service level requires additional memory resources compared to pure IaaS virtualization. In the remainder of this section we quantify and analyze these overheads.

Cryptographic Primitives

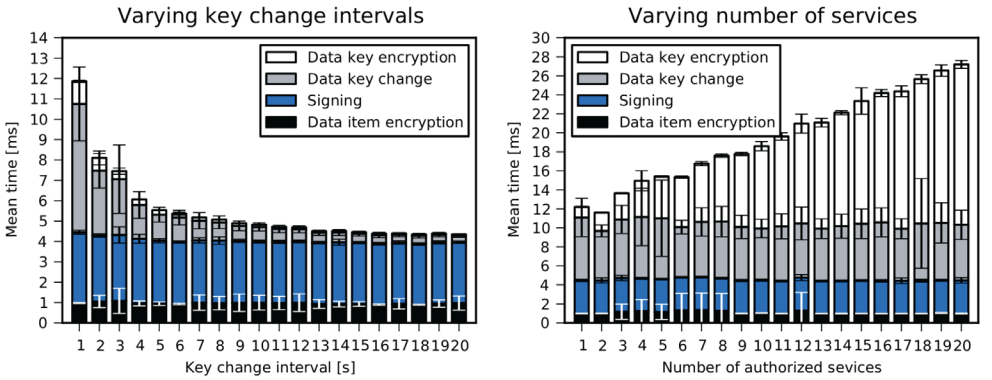
We use cryptographic primitives with at least 112-bit security for symmetric operations in order to provide data security up to the year 2030 according to NIST (Barker et al., 2012). The choice of cryptographic primitives was further guided by reducing the computational burden on the Trust Point. Constantly, the Trust Point has to perform three different 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 data field encryption, we use AES with 128 bit keys in CBC mode. As the AES operations only lead to a marginal overhead, we limit or analysis to sensor data containing only a single data field, unless specified otherwise explicitly. For data item signing, we use the ECDSA scheme with NIST curve P-224. Finally, for encrypting data encryption keys for authorized services we use RSA with 2048 bit keys. This choice was mainly guided by the advantageous performance asymmetry for public-key operations, which lowers the burden on the Trust Point. For future revisions of our architecture we may replace RSA by elliptic curve encryption mechanisms such as ECIES.

Performance Overhead

There are extensive studies on the performance overhead of the transport security mechanisms employed by our architecture (Coarfa et al., 2006). Thus, we focus on evaluating our object security mechanisms. These introduce computational overhead for both Trust Point and Cloud. For the Trust Point, this overhead results from the per-data field encryption, the per-data item integrity protection, and the encryption of keys for authorized services. Correspondingly, services have to decrypt encrypted data fields before processing them and need to verify the integrity of data. In order to allow fine-grained data access control, the Trust Point periodically changes and distributes encryption keys. After each of these key changes, services have to decrypt the encryption keys with their private key. We measure this performance overhead using our OpenSSL-based prototype.

For each data point, we conducted 40 measurements. We show the mean processing time for one data item with one data field as well as the corresponding standard deviation. The left part of Figure 5 shows the results for the processing time of a data item with one authorized service and increasing key change intervals. The results show that even for a key change interval of 1 second, our low-end Trust Point is able to process more than 80 data items per second. For a key change interval of only 10 seconds, this value increases to 200 data items per second. The figure also shows that for key change intervals larger than 10 seconds, the integrity protection accounts for more than 75 percent of the processing time. Thus, further increasing the key change interval reduces the per-data item processing time on the Trust Point only marginally. The right part of Figure 5 shows the results for processing one data item with a key change interval of 1 second for an increasing number of authorized services. For

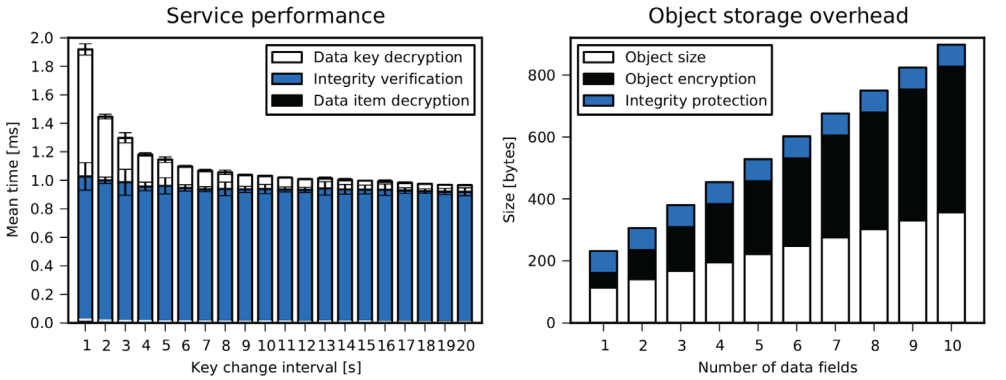
Figure 5. The mean time for processing one data item depends on the key change interval (left, one authorized service) and the number of authorized services (right, key change interval 1 s)



each additional service, the Trust Point has to encrypt the encryption key on every key change. The results show that even with a key change interval of 1 second and ten authorized services, the Trust Point is able to process more than 50 data items per second. For an increasing number of authorized services, the data key encryption becomes the main bottleneck. To bring these numbers into perspective, assume a per-sensor sampling rate of one measurement per second and ten authorized services per sensor node. Even in this exaggerated scenario, the low-end Trust Point is able to protect sensor data for over 50 nodes. In order to support higher sampling rates or larger network sizes, the hardware of the Trust Point can easily be scaled up or hardware support for cryptographic algorithms can be added. Thus, throughputs that are one to two orders of magnitude higher can easily be achieved. The main limiting factor for protecting sensor data at the Trust Point are the public key operations for the data integrity mechanism and key distribution. Fortunately, modern multicore SoCs allow to perform these calculations at higher core speeds and in parallel compared to the single core architecture of the Raspberry Pi. Our evaluation shows that the number of data fields per data item only has a marginal influence on the maximum throughput of the Trust Point. The symmetric encryption used to encrypt data fields can be computed efficiently in 0.004 ms on the Raspberry Pi. Hence, our architecture smoothly scales with the number of data fields.

Considering Cloud services, we are interested in the number of data items a service can decrypt and verify in one second. The left part of Figure 6 depicts the mean processing time for decrypting and verifying one data item for increasing key change intervals. Even for a key change interval of 1 second, the service is able do decrypt 500 data items per second. For a key change interval of 20 seconds, the throughput increases to nearly 1,000 data items per second. Again, the integrity protection leads to a constant performance overhead. In order to reduce this overhead, we propose two approaches, probabilistic verification and verification-as-a-service. If a service trusts the Cloud provider to a certain extend, the service can use *probabilistic verification* to only verify the integrity of a random sample and thus dramatically increase throughput. First results show that throughput can be increased up to 9,900 data items per second for a verification probability of 10 percent and even up to 96,300 data items per second for a verification probability of 1 percent (Hummen et al., 2012). However, if deterministic verification is required (i.e., when using an untrusted Cloud provider), the data owner may use verification-as-a-service which can be offered by dedicated services that

Figure 6. The overhead consists of processing sensor data in a service (left, varying key change intervals) and storing encrypted sensor data (right, increasing number of data fields)



continuously verify the integrity of all stored data (Wang et al., 2011). Overall, the results of the performance overhead evaluation show that our security design scales well for our intended scenario.

Storage Overhead

We identified two kinds of storage overhead that need to be evaluated. The first type of overhead follows directly from the object security mechanisms, i.e., data encryption and integrity protection. The second type results from the storage of the encrypted keys in the Cloud.

We consider a simple JSON-encoded sensor data item with four (unencrypted) meta data fields and an increasing number of measured values that have to be encrypted. The right part of Figure 6 depicts this storage overhead of our proposed object security mechanisms for an increasing number of (encrypted) data fields. The constant overhead for the JSON-encoded meta data accounts for 87 bytes and each measured value adds additional 27 bytes (assuming that data fields fit into a single 16 byte block). For each data field, the overhead consists of a 16-byte initialization vector and an additional overhead of 31 bytes for the JSON-encoded JWE information. Additionally, the integrity protection checksum accounts for a constant overhead of 64 bytes plus 7 bytes for its JSON encoding. Consequently, the cryptographic storage overhead grows linearly with the number of data fields in each data item. For a reasonable amount of data fields, this overhead is well manageable with the elastic storage resources offered by today's Cloud solutions.

For each service that is authorized to access a data item, the Cloud stores the encryption keys of protected data items in its dedicated key store. Thus, the overhead for key storage in the Cloud strongly depends on the key change interval, the number of connected sensors, and the number services authorized to access a sensor's data. Table 1 shows the calculated storage overhead per sensor and data field for varying key change intervals and number of services for one month of sensor data (assuming a sample rate of once per second). For a reasonable key change interval, the key storage overhead approaches practical sizes, e.g., 70 MB per month at a per minute granularity for each authorized service. Additionally, the data owner may delete or aggregate old data in order to decrease the total storage overhead in the Cloud. In conclusion, the storage overhead imposed by the security architecture is well manageable using the elastic storage capabilities of the Cloud.

SEE Memory Overhead

To evaluate the memory overhead imposed by the SEEs for service isolation, we developed an early SEE prototype based on Linux Secure

Table 1. Key storage overhead with respect to key change interval and authorized services

		Key Change Interval					
		1 Second	**1 Minute**	**1 Hour**	**1 Day**	**1 Week**	**1 Month**
Services	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

Containers (LXC). Inside the LXC container, an OpenJDK JVM executed a minimal Java test service creating some I/O load every few seconds. Each service was executed in its own LXC container, which guaranteed secure isolation of the processes (i.e., the JVMs) and which allowed to define quotas for processing, memory, and I/O usage. The memory consumed by a single SEE instance and the test service amounted to roughly 7 MiB. Consequently, we were able to launch more than 1000 SEEs in parallel on one EC2 large instance with 7.5 GiB of RAM. We did also not observe any issues with the kernel or scheduler due to the high number of containers, i.e., all services were able to perform their I/O operations as desired. Today's IaaS Clouds can easily provide instances with more than 7.5 GiB of RAM. For example, Amazon's High Memory Cluster Eight Extra Large-Instance comes with 244 GiB of RAM (Amazon Web Services, 2013), which equates to more than 32,000 SEEs each executing a different service at the desired level of separation on a single physical machine.

CONCLUSION

Multiple, possibly unknown or untrusted, stakeholders are involved when outsourcing storage and processing of sensor data to the Cloud. In this paper, we discussed threats originating from these stakeholders. We counter these threats with a security architecture that enables the data owner to stay in control over her data. For this purpose, we introduce a Trust Point as a new logical entity. It is located at the sensor network's border and acts as a bridge between the trust domains of the sensor network and the Cloud. Using the Trust Point, we i) implement transport security mechanisms for secure communication with the Cloud, ii) apply object security mechanisms to sensor data sent to the Cloud, and iii) perform key management in order to authorize services. Additionally, we suggest using isolation mechanisms for services, which mitigates the leakage of sensitive information from the run-time contexts of services. Our evaluation validates an adequate performance of our security architecture for the intended scenario and shows that the introduced storage and memory overheads can be handled effectively. Thus, user-controlled storage and processing of sensor data in the Cloud is a promising extension of today's Cloud offers.

ACKNOWLEDGMENT

The authors would like to thank all partners of the SensorCloud consortium for 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.

This is an extended and completely rewritten paper based on the work *A Cloud Design for User-controlled Storage and Processing of Sensor Data* presented at IEEE CloudCom 2012 (Hummen et al., 2012).

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). A survey on sensor networks. *IEEE Communications Magazine*, *40*(8), 102–114. doi:10.1109/MCOM.2002.1024422

Amazon Web Services. (2013). *Amazon EC2 instance types*. Retrieved April 10, 2013, from http://aws.amazon.com/en/ec2/instance-types/

Barker, E., Barker, W., Burr, W., Polk, W., & Smid M. (2012). *Recommendation for key management – Part 1: General (Revision 3)*. National Institute of Standards and Technology, NIST Special Publication 800-57.

Becker, E., Buhse, W., Günnewig, D., & Rump, N. (Eds.). (2003). *Digital rights management*. Springer. doi:10.1007/b12637

Boneh, D., & Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Proceedings 4th Theory of Cryptography Conference (TCC 2007)*.

Bugiel, S., Nürnberger, S., Sadeghi, A.-R., & Schneider, T. (2011). Twin clouds: Secure cloud computing with low latency. In *Proceedings 12th Joint IFIP TC6 and TC11 Conference Communications and Multimedia Security (CMS 2011)*.

Chow, R., Golle, P., Jakobsson, M., Shi, E., Staddon, J., Masuoka, R., & Molina, J. (2009). Controlling data in the cloud: Outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW 2009)*.

Coarfa, C., Druschel, P., & Wallach, D. S. (2006). Performance analysis of TLS web servers. [TOCS]. *ACM Transactions on Computer Systems*, *24*(1), 39–69. doi:10.1145/1124153.1124155

Crockford, D. (2006). *The application/json media type for JavaScript object notation (JSON). Internet Engineering Taskforce RFC 4627*. Informational.

Danezis, G., & Livshits, B. (2011). Towards ensuring client-side computational integrity. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security (CCSW 2011)*.

Federal Office for Information Security. (2011). *Protection profile for the gateway of a smart metering system*. Germany, v01.01.01(final draft).

Galiegue, F., Zyp, K., & Court, G. (2013). *JSON Schema: Core definitions and terminology*. Internet Engineering Taskforce Internet-Draft draft-zyp-json-schema-04 (work in progess).

Gentry, C. (2010). Computing arbitrary functions of encrypted data. *Communications of the ACM*, *53*(3), 97–105. doi:10.1145/1666420.1666444

Geoffray, N., Thomas, G., Muller, G., Parrend, P., Frénot, S., & Folliot, B. (2009) I-JVM: A Java virtual machine for component isolation in OSGi. In *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*.

Hammer-Lahav, E. (2010). *The OAuth 1.0 protocol. Internet engineering taskforce RFC 5849*. Informational.

Henze, M., Hummen, R., & Wehrle, K. (in press). The cloud needs cross-layer data handling annotations. In *Proceedings of the 4th International Workshop on Data Usage Management (DUMA 2013)*.

Hummen, R., Henze, M., Catrein, D., & Wehrle, K. (2012). A cloud design for user-controlled storage and processing of sensor data. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2012)*.

Itani, W., Kayssi, A., & Chehab, A. (2009). Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures. In *Proceedings 8th IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC 2009)*.

Jones, M., Rescorla, E., & Hildebrand, J. (2012). *JSON web encryption (JWE)*. Internet engineering taskforce internet-draft draft-ietf-jose-json-web-encryption-08 (work in progess).

Kamara, S., & Lauter, K. (2010). Cryptographic cloud storage. In *Proceedings Workshop on Real-Life Cryptographic Protocols and Standardization (RLCPS 2010)*.

Kissner, L., & Song, D. (2005). Privacy-preserving set operations. In *Proceedings 25th Annual International Cryptology Conference (CRYPTO 2005)*.

Lombardi, F., & Di Pietro, R. (2011). Secure virtualization for cloud computing. *Journal of Network and Computer Applications*, *34*(4), 1113–1122. doi:10.1016/j.jnca.2010.06.008

Martignoni, L., Poosankam, P., Zaharia, M., Han, J., McCamant, S., & Song, D. … Stoica, I. (2012). Cloud terminal: Secure access to sensitive applications from untrusted systems. In *Proceedings of the 2012 USENIX Annual Technical Conference (ATC 2012)*.

Parrend, P., & Frenot, S. (2009). Security benchmarks of OSGi platforms: Toward hardened OSGi. *Software, Practice & Experience*, *39*(5), 471–499. doi:10.1002/spe.906

Pearson, S., & Benameur, A. (2010). Privacy, security and trust issues arising from cloud computing. In *Proceedings of the Second IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2010)*.

Pearson, S., Mont, M., Chen, L., & Reed, A. (2011). End-to-end policy-based encryption and management of data in the cloud. In *Proceedings of the Third IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*.

Popa, R. A., Redfield, C. M. S., Zeldovich, N., & Balakrishnan, H. (2011). CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles (SOSP 2011)*.

Ristenpart, T., Tromer, E., Shacham, H., & Savage, S. (2009) Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*.

Santos, N., Gummadi, K. P., & Rodrigues, R. (2009). Towards trusted cloud computing. In *Proceedings USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 2009)*.

Sehr, D., Muth, R., Biffle, C., Khimenko, V., Pasko, E., & Schimpf, K. … Chen, B. (2010). Adapting software fault isolation to contemporary CPU architectures. In *Proceedings of the 19th USENIX Conference on Security (Security 2010)*.

van Dijk, M., & Juels, A. (2010). On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proceedings 5th USENIX Workshop on Hot Topics in Security (HotSec 2010)*.

Wallom, D., Turilli, M., Taylor, G., Hargreaves, N., Martin, A., Raun, A., & McMoran, A. (2011) myTrustedCloud: Trusted cloud infrastructure for security-critical computation and data management. In *Proceedings of the Third IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*.

Wang, Q., Wang, C., Ren, K., Lou, W., & Li, J. (2011). Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, *22*(5), 847–859. doi:10.1109/TPDS.2010.183

Zhang, Y., Juels, A., Reiter, M. K., & Ristenpart, T. (2012). Cross-VM side channels and their use to extract private keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*.

CALL FOR ARTICLES

International Journal of Grid and High Performance Computing

An official publication of the Information Resources Management Association

MISSION:

The primary mission of the **International Journal of Grid and High Performance Computing (IJGHPC)** is to provide an international forum for the dissemination and development of theory and practice in grid and cloud computing. IJGHPC publishes refereed and original research papers and welcomes contributions on current trends, new issues, tools, societal impact, and directions for future research in the areas of grid and cloud computing. This journal is targeted at both academic researchers and practicing IT professionals.

ISSN 1938-0259
eISSN1938-0267
Published quarterly

COVERAGE/MAJOR TOPICS:

- Advanced collaboration techniques and scaling issues
- Algorithms and techniques for HPC
- Big Data
- Bio-inspired grid resource management
- Cloud architectures
- Cloud business process integration
- Cloud client and applications
- Cloud engineering and management
- Cloud foundation concepts
- Cloud Platforms and Infrastructures
- Cloud reliability and security
- Cloud Services
- Cloud standards
- Cloud types
- Combating global terrorism with the world wide grid
- Emerging standards for organizations and international projects
- Future of grid, trends, and challenges
- Green data centers
- Grid and software engineering aspects
- Grid architecture, resources, and data management
- Grid economy, market dynamics, and simulations
- Grid education and applications - science, engineering, and business
- Grid evolution, characterization, and concepts
- Grid fundamentals, algorithms, and performance analysis
- Grid impact, scientific, and industrial and social implications
- Grid instrumentation, measurement, and visualization
- Grid middleware, scheduling, brokering, and monitoring
- Grid portals and security
- Grid programming, models, tools, and API
- Grid services, concepts, specifications, and frameworks
- Grid uses and emerging technology
- New initiatives, SOA, autonomic computing, and semantic grid
- Simple API for grid applications (SAGA)
- Software and hardware support for HPC
- Test, evaluation, and certificate presentation
- Wireless and optical grid, characteristics, and applications
- Work flow management

All inquiries regarding IJGHPC should be directed to the attention of:
Emmanuel Udoh, Editor-in-Chief
udohe123@yahoo.com
All manuscript submissions to IJGHPC should be sent through the online submission system:
http://www.igi-global.com/authorseditors/titlesubmission/newproject.aspx

Ideas for Special Theme Issues may be submitted to the Editor-in-Chief.

Please recommend this publication to your librarian. For a convenient easy-to-use library recommendation form, please visit:
http://www.igi-global.com/IJGHPC