The International Journal of Parallel, Emergent and Distributed Systems
Vol. 00, No. 00, Month 2017, 1–16

RESEARCH ARTICLE

Fast-Sec: An Approach to Secure Big Data Processing in the Cloud

Julio C.S. dos Anjos[1], Tatiana Galibus[2], Cláudio Resin Geyer[1], Gilles Fedak[3], João Paulo C. L. Costa[4], Rubem Pereira[5], Edison Pignaton de Freitas[1]

[1]*Federal University of Rio Grande do Sul, Institute of Informatics - PPGC - Brazil*
{jcsanjos, geyer, edison.pignaton }@inf.ufrgs.br;
[2]*Belarusian State University, 220030 Minsk, 4, Nezavisimosti – Minsk – Belarus*
tan2tan@gmail.com
[3]*INRIA, Avalon - ENS Lyon, France*
gilles.fedak@inria.fr
[4]*University of Brasilia, UnB - FT – ENE – CP: 4386 - 70910-900, Brasília - DF - Brazil*
joaopaulo.dacosta@ene.unb.br
[5]*Liverpool John Moores University, LJMU City Campus, Liverpool, England*
r.pereira@ljmu.ac.uk

(*v1.0 released April 2017*)

Security is an important concern in computer systems, which is especially remarkable when the system has to handle large amounts of data and a number of different users accessing this data with different accessing permissions. Observing this demand, this work proposes an innovative approach to provide a security infrastructure support to Big Data Analytic in Cloud-based systems named Fast-sec. Fast-Sec handles systems with large volumes of data from heterogeneous sources, in which users may access the system by different platforms, consuming or providing data. The security infrastructure proposed in Fast-Sec provides an authentication mechanism for users and data access control adapted to high demands from cloud-based Big Data environment. The reported results show the adequacy of the proposed safety infrastructure to the cloud-based systems processing Big Data.

Keywords: Big Data, MapReduce, Distributed System, Hybrid infrastructures, Security for Hybrid System

1. Introduction

Computer systems can collect and store data about different human interests and follow its evolution almost on a continuous basis. Similarly, the environment (built and natural) is ever more monitored by sensing devices that sample some aspects of the environment and generate large volumes of data. This is a fact that pushes strong demands to data processing and storage, particularly if real-time requirements have to be observed [1]. Observing this reality, cloud computing is a promising technology to address the challenges imposed by these new demands [2]. It already is an essential part of the global IT reality and it has been used as a platform for business applications and data processing [3]. The current challenge is how to construct such efficient processing infrastructures to handle massive amounts of data, which is a great demand in the field of Big Data research and industry communities.

Due to fact that this approach has specific characteristics which distinguish it from other applications, such as, volume, variety, velocity, value and veracity [4] it demands

[1]Corresponding author. Email: jcsanjos@inf.ufrgs.br

new solutions. A large volume of data (volume) from different sources that ingress in the cloud with different formats and sources (variety) are processed in real-time (velocity) with high accuracy (veracity) to find competitive information (value) for society. However, data need to be homogenized and coherently handled by the cloud or other infrastructures, like hybrid infrastructures [5]. This infrastructure needs to be secured to guarantee the data privacy and avoid miss usage of the services it offers.

The key vulnerabilities of such infrastructures are the constantly increasing openness and scalability. The proposed solutions are focused mainly on achieving the basic Big Data processing characteristics. The security requirements for such systems are not taken into account and the solutions are not analyzed from the point of security [6]. The importance of security increases rapidly with the growth of Big Data industry. The efficient Big Data security infrastructure should include the following components: Authentication services; Access control infrastructure for the supported access policies; Encryption protocols and other standard cryptography services; Fast on-the-fly auditing services [7].

Currently, there are several industrial approaches to securing the Big Data infrastructures on the market, such as: Cloudera Sentry, IBM InfoSphere Optim Data Masking, Intel's secure Hadoop distribution, Protegrity Big Data Protector for Hadoop, Revelytix Loom,Zettaset Secure Data Warehouse, Apache Accumulo and other. All mentioned approaches are designed for the Hadoop infrastructure and utilize Kerberos secure resource sharing protocol. In the theoretical field it is possible to mention some works related to the concept of access control policy [8], hybrid environment machine-learning systems [9] among other, but there are no general analytical articles related to Big Data security. The existing approaches to securing the Big Data processing systems are designed and implemented as patches, *i.e.*, adjusted to the current needs of the system. The Big Data security theory lacks the formal conceptual models of the consistently secure Big Data processing systems, which obviously leads to the absence of the verifiable solutions for security [10] and [11].

The top 10 security problems in the Big Data cloud processing are mentioned in the CSA (Cloud security alliance) paper [12]. *Fast-Sec* presents a proposal to address one of the most demanding of these problems, *i.e.*, the problem of access control and which is related to the user privacy protection. The proposal presents a novel approach to the Big Data access control implementation. The methodology is based on the specific access control methods designed for the Big Data processing incorporated into the cloud infrastructure. The proposal includes a framework called Small & Medium sized Enterprise Data Analytic in Real Time (SMART) [13] and related access control methods that support and protect the data analytic processes and associated access policy. SMART is a modular framework for Big Data analysis that takes advantage of cloud, multiple clouds or hybrid infrastructures to provide support for Small & Medium-sized services operation. The core access control framework has already been successfully implemented in the protected cloud infrastructure. The complete solution provides a secure and flexible cloud-based Big Data processing system capable to support the different types of services that can be securely combined to address specific needs of different application domains with different level of user access.

To describe the proposed Fast-Sec approach, its contributions and positioning in relation to related work, this paper is structured as follows: Section 2 examines the works that are related with this proposal. In Section 3, there is an overview description of the main characteristics of the proposed cloud infrastructure. Section 4.2 describes the proposed authentication and access control mechanisms for this cloud infrastructure. Section 5 presents the acquired results for these proposed security mechanisms. Section 6 concludes the paper and presents suggestions for future work.

2. Related Work

The data of the organizations might involve both sensitive and non-sensitive information. The computation of this profile should not be carried out in the public cloud without a safe environment. Many users and businesses hesitate to keep confidential data in a CSP because of data integrity and security problems, as described by Zhang [14]. The study of Zhang entails conducting a secure search of distributed keywords across multiple clouds. The aim is to provide a secure search service for encrypted cloud data. This approach has two schemes: the first scheme is a *cross-store*, *i.e.* all *file slices*, keywords and keys are encrypted (this provides efficiency and anonymity for data owners). The second scheme involves constructing a *distribution strategy* for both keyword and files (this sets out the search and security requirements). These schemes are combined with *Shamir's secret scheme* to improve the distributed search system, where, all the files and keywords for different CSPs are encrypted with different secret keys.

In the *cross-store*, each file is encrypted through an encryption key before it is distributed to N cloud servers or for some server in a particular CSP. The user has one particular secret key $H_s(k_{w,j})$ which will be used to encrypt a file segment, such as $C_{i,j} = (F_{i,j})_{H_s(K_{f,j})}$. The file distribution is generated from random *file slices* $(F_{i,1}, F_{i,2}, \ldots, F_{i,N-1})$ with the same length. Each file F_i carries out an exclusive OR operation $\oplus$ and $F_{i,N} = F_{i,1} \oplus F_{i,2} \oplus \ldots \oplus F_{i,N-1} \oplus F_i$. Then the secret keys ($k_{f,j}$, $k_{w,j}$) and *file slices* ($F_{i,N}$) are cross-stored among these CSPs. The authorized user retrieves the *file slices* and the secret keys when he wants to access the data directly.

In the *distribution strategy*, a secret key is used to encrypt the files and a secret key matrix shows the order of iterative encryption in the *file slices*. The file distribution is stored in three stages: a) Each file is encrypted and partitioned into N *file slices*. b) The system creates an iterative encryption matrix M with the *file slices* that are stored with the user. c) The encrypted file is distributed over CSPs without the providers knowing the secret matrix. When an authorized user wants to execute a security search, he issues a *trapdoor* to CSPs to retrieve the encrypted file slices. Following this, these file slices are decrypted and the original file is reconstructed. The *trapdoor* is an encrypted keyword without previous information from the destination. The computational complexity is $O((N!)^T)$, where N is the number of CSPs and T is the number of iterative encryption rounds.

Hybrid clouds uses local resources to process a determined number of workloads. When these resources dry up in a "have-tail consume", more availability is needed in a public cloud, and issues like security and privacy become a problem, as shown in the study of [15]. A safe computational approach to *hybrid cloud* is to separate sensitive from non-sensitive data. The authors introduce a framework to *MapReduce* for secure computing with mixed-sensitivity data in *hybrid cloud* called *Tagged-MapReduce*. The data receives sensitive or non-sensitive flags and the key/value pair has an equivalent tag.

The goal is to prevent sensitive data from leaving private cloud. The programmers can decide where the computation will be executed. As a result, the input data and the intermediate keys can be moved from a private cloud to a public cloud without compromising security. The *Map* and *Reduce* phases produce {key/value; tag} pairs when a sensitive input data is received. The scheduling of both HDFS and *Jobtracker* will identify the tag. The function of the tag is to indicate the execution path for the system. The scheduler employs all the intermediate results for private cloud to carry out the reduction, even though the intermediate transfers might overload the private servers and generate a high volume of data from clouds (whether public or private).

The Cloud Security Alliance report [12] describes the most important problems of securing the distributed data processing systems. Among them it is possible to highlight: mapper security, real-time monitoring, end-point security, middleware protection, user privacy protection and access control. Traditional and conventional security mechanisms

are not suitable to be direct applied to the emerging distributed services present in current cloud-based systems, and adapting them to this new context is a very challenging task. The critical components, such as access control, remain elusive [16], [17].

In line with the above highlighted aspect, it is possible to take Hadoop-oriented security approaches based on the Kerberos resource sharing protocol [18], which turn out to be ineffective in relation to support and implementation [17], [16]. Besides that, it is difficult, and rather time-consuming, to set up and configure the Kerberos infrastructure [16]. In cloud computing systems, it is preferable to use either public-key infrastructure (PKI) or Attribute-based Encryption (ABE) for protecting the user privacy [19], while the symmetric encryption should be used to support the fast processing of the bulk data [20].

Scrutinizing the literature, it is possible to observe that apart from the Hadoop security infrastructure based on Kerberos, there is a wide demand for developing security models and policies in for the cloud domain [16], [21], [22], [7]. Such models allow to analyze the consistency of a proposed security architecture, and figure out the basic adversary model [21]. Other can be applied for distributed data processing infrastructures, such as Flink, possessing more flexibility in providing various data processing functions [5].

According to the CSA report [12], the most promising cryptographic tool for security framework implementation is the Attribute-based Encryption (ABE) [23], [24], [25]. The most important challenge related to ABE implementation is the necessity to apply modifications in order to support the key revocation [26] and simplify the key generation [19].

In light of these problems, this current paper reports the work to provide a security model corresponding to the attribute-based access model, with a corresponding implementation design to support the model. The goal is not only to propose the model as in related work, [16], [21], [22], but also to provide a practical ABE-based framework design and implementation.

3. Cloud infrastructure

This Section shows the main features and an overview of the infrastructure behind from safety approach of Smart-Sec described in the Section 4.2. The infrastructure proposed is the core to the environment in the Cloud with different devices following an attribute access-based policy. Thus, it is a Big Data processing services in the SMART cloud-based architecture.

3.1 Architecture Overview

Different cloud infrastructures have their own configuration parameters, and the availability and performance of offered resources can change dynamically according to several factors, including the degree of over-commitment that a provider employs. In this context, solutions are needed for the automatic configuration of complex cloud services at several abstraction levels. Cloud infrastructures comprising heterogeneous hardware environments may need the specification of configuration parameters at several levels such as the operational system, service containers and network capabilities [27]. As users need to execute applications may not know how to map their requirements to available resources, this lack of knowledge about the cloud provider infrastructure will lead either to overestimation or underestimation; both are equally bad as the former leads to waste of resources whereas the second sacrifices quality of service.

The application profile is optimized for all file sizes in the available infrastructure. As the systems are independent different data sizes can be manipulated at same time. Previous work has been performed on *MapReduce* for hybrid environments [5], [28]. Figure 1 illustrates the solution proposed to model a hybrid system which depicts a Global Dis-

patcher and Global Aggregator to be used on the infrastructure.

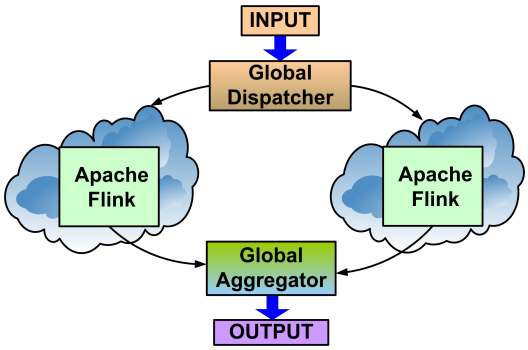

Figure 1. SMART Architecture

The Global Dispatcher located outside the cloud has middleware functions for handling task assignment, and management of user-provided data. It is a centralized data storage system that manages policies for split data and distribution in accordance with the needs of each system. The working principle is similar to a publish/subscribe service (PS-S) in which the system obtains data and publishes computing results [5]. PS-S is a mechanism producer/consumer that works like a *Queue Message*. The user defines tasks and local data for put in a queue over Apache Kafka in similar manner to [29] and after the scheduler from global dispatcher distributes this tasks in the queue for machines in the Cloud. The Global Aggregator data output from both systems and merges them in order to obtain the final data set.

3.2 SMART Internal Infrastructure Components

Hybrid infrastructure, where there are many cloud providers with heterogeneous environments and configurations, often needs to use an orchestrator to manage the results and data input from users. The orchestrator must be decentralized in order to improve data distribution in the network. The infrastructure enables the use of highly heterogeneous machines. When considering the use of a public cloud to extend the capacity of a community cloud, or desktop grid, several scenarios and data strategies are possible. The extent to which a set of data-distribution strategies is applicable to a given scenario depends on how much bandwidth is available. Two distinct Distributed File System (DFS) implementations may be required to handle data distribution in two scenarios, namely low-bandwidth and high-bandwidth. The Apache Flink, formerly known as Stratosphere [30], is the base infrastructure of the SMART framework system, represented in Figure 2. Its flexible pipeline enables several *MapReduce* and extended functions like Map, MapPartion, Reduce, Aggregate, Join and Iterative. It can be used in order to allow this cloud extension. The setting will be transparent to users because a middleware in a top level abstracts the complexity away from the users. The different layers of the stack build on top of each other and raise the abstraction level of the program representations they accept:

- The API layer implements multiple APIs that create operator DAGs for their programs. Each API needs to provide utilities (serializers, comparators) that describe the interaction between its data types and the runtime. All programming APIs are translated to an intermediate program representation that is compiled and optimized via a cost-based optimizer.
- The Flink Common API and Optimizer layer takes programs in the form of operator DAGs. The operators are specific (e.g., Map, Join, Filter, Reduce, FlatMap, MapPartition, ReduceGroup, Aggregate, Union, Cross, etc) and the data is in nonuniform type.

The concrete types and their interaction with the runtime are specified by the higher layers.

- The Flink Runtime layer receives a program in the form of a JobGraph. A JobGraph is a generic parallel data flow with arbitrary tasks that consume and produce data streams. The runtime is designed to perform very well both in setups with abundant memory and in setups where memory is scarce.

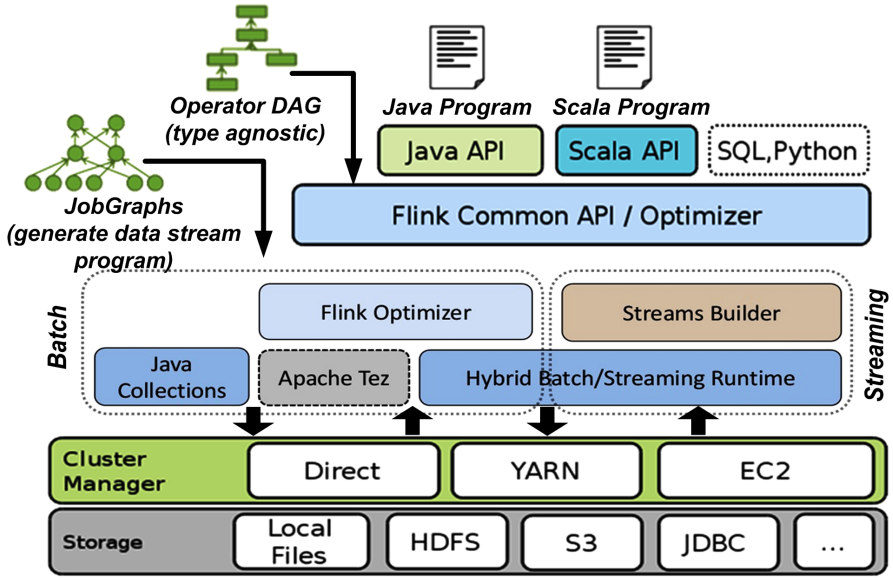

Figure 2. Apache Flink component stack

Flink explores the power of massively parallel computing for advanced analytics and leverages a novel, database inspired approach to analyze, aggregate, and query very large collections of either textual or (semi-)structured data on a virtualized, massively parallel cluster architecture. It combines the strengths of MapReduce/Hadoop with powerful programming abstractions in Java and Scala and a high performance runtime. In addition to basic data flow concepts as common in relational databases, or systems like Hadoop and Spark, Flink has native support for iterations, incremental iterations, and programs consisting of large DAGs (Directed Acyclic Graph) of operations. Particularly, it is possible to highlight the following:

- Flink uses a richer set of primitives than MapReduce, including primitives that allow the easy specification, automatic optimization, and efficient execution of joins. This makes the system a more attractive compilation platform for data warehousing, information extraction, information integration, and many other applications. The programmer does not need to worry about writing parallel code or hand-picking a join order.
- Flink includes native support rather than outside loop in Mahout [31] on top of Hadoop for iterative programs that make repeated passes over a data set updating a model until they converge to a solution. Flink contains explicit “iterate” operators including bulk iteration and delta iteration that enable very efficient loops over data sets, e.g., for machine learning and graph applications. These operators enable the specification, optimization, and execution of graph analytic and statistical applications inside the data processing engine.
- Different from Spark, Flink uses an execution engine that includes external memory query processing algorithms and natively supports arbitrarily long programs shaped as DAG. Flink offers both pipeline (inter-operator) and data (intra-operator) parallelism. Flink’s runtime is designed as a data processing pipeline engine rather than a batch

processing engine, thus it supports both batch and streaming processing. Operators do not wait for their predecessors to finish in order starting processing data. This results in a very efficient handling of large data sets.

4. The Proposed Fast-Sec Security Model

The proposed approach to secure Big Data systems on the cloud is an evolution of a cloud-based access control and privacy protection infrastructure which is currently implemented in protected enterprise cloud storage - Storgrid [32]. This infrastructure is the core of the protected environment in the cloud with the heterogeneous devices and attribute access policy. Therefore, it is suitable for the SMART cloud-based architecture and corresponding Big Data processing services. The proposed security model tackles authentication and access control problems with a modular and light-weight approach that will be have its components first detailed and then explained how it is integrated in the SMART.

4.1 Authentication and Access Control Mechanisms

In order to address the issues related to authentication and access control, the proposed approach consists in an infrastructure including the following components: Encryption server; File storage; Key storage; and Client UI. These elements are illustrated in Figure 3 and detailed in the following.

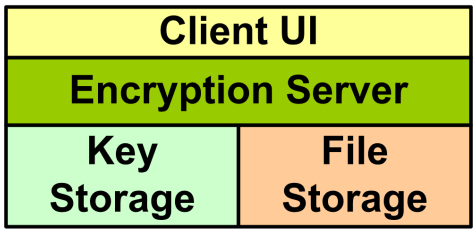

Figure 3. Security Infrastructure Components

Encryption server manages all the certified authentication (CA) and encryption operations and grants the user access to the data storage. This server can store the encryption keys and/or connect to a separate Key Storage server.

File storage is secure in the sense that some of the files specified by the domain administrator are store, encrypted and have restricted access. Due to the fact that the file storage data is partially stored in the cloud, *i.e.*, externally it is recommended to encrypt this external part of file storage completely.

Client UI can connect to the *Encryption server* and ask for the permission to access the file storage in order to view/edit/upload specific files or folders.

All components of the proposed system support the high-level cryptographic functions, including different types of encryption, timestamp verification and hashing. In order to increase efficiency, a hybrid encryption system is set up which is combined of both symmetric and attribute-based encryption. This configuration allows controlling the privacy of the users without compromising the overall encryption time. In addition, the basic ABE [23], [24] approach was modified to improve configuration parameters and increase the speed of encryption, for the purpose of set up the validation period from user key and more sophisticated attributes corresponding to both the file shares and user groups. Nevertheless, the detailed description of ABE encryption modification is out of the scope of this article. The basic functionality of the security components are briefly described in the following:

File storage: The bulk data in the protected file storage is encrypted with the appropriate block cypher (AES, Blowfish, IDEA, Serpent). The key to the encrypted data is stored in the key storage and has expiry period in order to increase the protection. Once the portion of data is sent to the client, it is decrypted by the server and re-encrypted with another unique user session key.

Key storage: The symmetric keys for the data in the file storage are kept in the separate storage. The protection of the key storage is implemented via some strong authentication method, *i.e.*, two-factor authentication. Additionally, in order to increase the security of the sensitive data we propose the following methods:

a. Set up the key expiry period;
b. Use separate key for the different files;
c. Use secret sharing mechanism to key storing with the most sensitive data.

Encryption server: The most important cryptography services are run on the Encryption server. This server generates the user keys and connects to the client UI, *i.e.*, a separate user of the system and decides whether the access to the specific dataset should be granted to this user. In addition, the server runs the key renewal routines, stores the user public keys and attributes besides the auditing data.

Client UI: The client UI connects to the encryption server and checks the expiry period of the user keys (in the case it has been configured) and permits the device user to view/edit/upload the data. Client UI stores the user keys for the ABE encryption and the unique symmetric session keys which serve for restricting the access to the downloaded files. The symmetric keys are encrypted with the ABE keys. The client allows the whole system to work in the heterogeneous environment as it supports different platforms and operating systems.

The described modular infrastructure allows setting up different components separately and configuring the security system according to specific needs. The integration of the security infrastructure into the Big Data environment is possible due to its flexibility and scalable architecture. The main purpose of the proposed security infrastructure is the setup of the access control in the cloud-based protected environment. The access control mechanism performs tasks as follows:

(1) *Authentication of users in the cloud system:* The initial authentication is performed by user password and email id. In order to increase the security hierarchical system of authentication to the highly sensitive data is implemented a more sophisticated two-factor authentication, apart from the password and access to e-mail, the possession of a specific device is verified too. This can be used in the government services or services with the highly sensitive data.
(2) *Provision of access control functions and protection of data from the unauthorized access:* once the user is authenticated he/she can get access to the functions of the storage in order to upload or to download the data. The CA services are run by the encryption server. The server generates and distributes the user keys and keeps the group attributes along with the file sharing ids. Access control allows to securely distribute and show to the user (or accept from user) only the data he is permitted to view/edit. In order to achieve this protection, a special version of ABE is used with the implementation of both possible policies: key policy – KP, and cyphertext policy – CP, in order to support the attributes of the groups of users as well as the attributes of the file shares. This algorithm is developed specifically for the access structure of the proposed cloud architecture.
(3) *Protection of of the user data privacy:* Once the user wishes to access a separate file downloaded on user device the client uses his/her ABE key after performing the authentication in order to decrypt the symmetric session key and open the file.

4.2 Security Model Details

The proposed approach to secure the distributed data processing on the cloud is an extension of a cloud-based access control and privacy protection infrastructure. The core of the protected environment in the cloud is the hybrid attribute-based encryption [19] with additional parameters which allow to support the protection of heterogeneous devices and attribute access policy avoiding the immense use/optimizing the use of computational resources. Therefore, it is suitable for the SMART cloud-based architecture and corresponding Big Data processing services.

A simpler method of securing the user-generated data is used for the services the require the rapid data processing. In other words, the Storgrid security framework is extended with the following infrastructure:

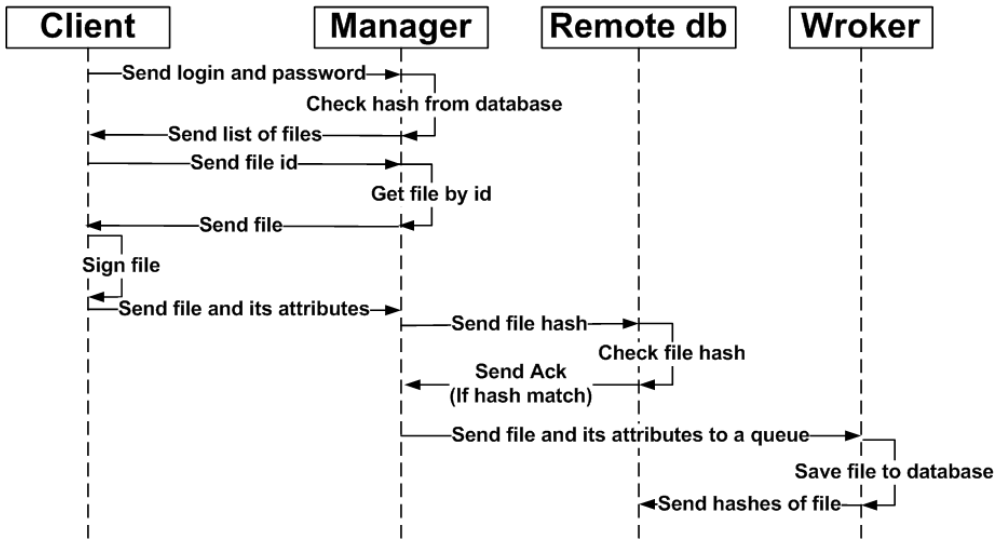

Figure 4. Smart Secure method

The protection of the services in such system is is based on the two basic mechanisms: 1) digital signature *i.e.* no one should be able to modify the data entered by a specific user other than the user himself and 2) the selective cryptography i.e. only the specific pieces of information collected are encrypted in order to improve the quality of sort and search service operations.

In order to protect the privacy and provide security for the enterprise services the initial Storgrid hybrid attribute-based AC is used. In this case, the performance is compromised for the sake of better security.

The proposed approach is based on several cryptographic mechanisms. The files/bulk data/multimedia are encrypted with 128/256-bit AES, while the permanent file keys are encrypted with the attribute-based encryption. The set of expiring ABE keys corresponding to the set of files accessible by user in encrypted with a single expiring AES key (KEY SET KEY). This key is split by server into 4 parts (2 are stored on the device and 2 belong to the user) by the secret sharing scheme (SSS). The encryption workflow is outlined in the following Figure 5.

With each user session, the permanent FILE KEY (unique AES key) is re-encrypted. The set of FILE KEYs is protected with the corresponding ABE keys. The unique ABE model supports the attribute policy based on user groups and on file shares. The model supports the simple selective ABE scheme [28], [29]. The selective scheme for attribute-based encryption is as follows: if at least one attribute in the set $\{t_i\}_U$ is equal to the attribute in the set $\{t_i\}_M$, the corresponding user U can decrypt the text M. In other words, as soon as the user and share have one attribute in common - the user can get

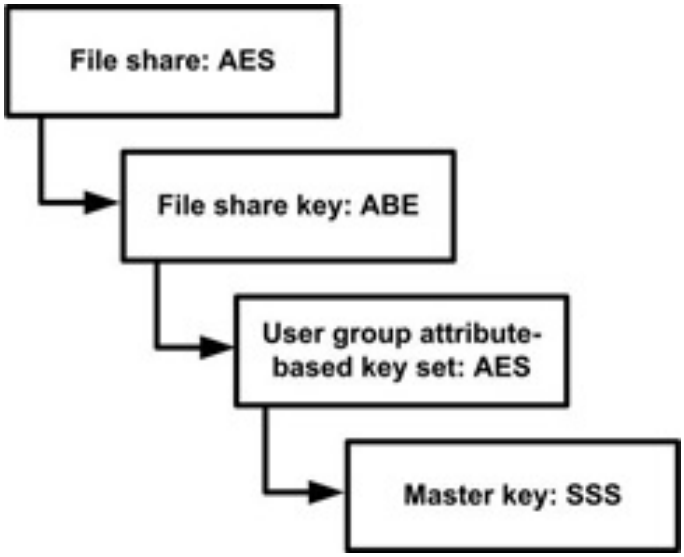

Figure 5. Encryption workflow

access to the share. The components of the ABE encryption are:

- Master-key (MK) which is kept safely on server and accessible only for the domain administrator and is specified by $MK = (t_1, t_2, ..., t_n, y)$, where the values t_i are randomly selected from the group Z_p. They are the private keys corresponding to the group attributes. Note, that this is different from the usual PK encryption: the private keys are controlled by the admin and not by the users;
- Public key (PK) depends on the master key values and is kept in the clear allowing users to access the information: $PK = (g^{t_1}, g^{t_2}, ..., g^{t_n}, .., e(g, g)^y)$, Here $e(g, g)$ is the bilinear pairing function corresponding to an elliptic curve;
- Secret user KEY SET depends on his attribute set. Here each D_i (GROUP KEY) serves for decryption of the data of a single group of users, for example, related to some project: $\{t_i\}_U \rightarrow D = \{D_i = g^{yw/t_i}\}$;
- Encrypted text M, in our context, $M =$ FILE KEY, or the permanent AES symmetric key, which allows to avoid the file re-encryption.

Encryption procedure is multiplication. The set of the public keys E_i (PUBLIC SHARE KEY) corresponding to the set of groups able to access the text is kept along with the encrypted text E:

$$E = Me(g, g)^y s, \{E_i = g^{t_i s/w}\}, i \in \{t_i\}_M$$

Decryption is division: $M = E/Y^s$

In order to perform this operation the user needs the pair of private key D_i and public key E_i corresponding to the attribute t_i :

$$Y^s = e(g, g)^{ys} = e(E_i, D_i) = e(g^{yw/t_i}, g^{t_i s/w}) = e(g, g)^{ys}$$

The result of decryption is the FILE KEY - the symmetric AES key that permits to decrypt the contents of protected file.

4.3 Security Infrastructure of the SMART Cloud-Based Processing Service

In order to combine the SMART architecture and the proposed hybrid CA system it is necessary to separate the functions of the encryption server, *i.e.*, the protection services that work once the data is uploaded (authentication, CA, encryption) which must run before the data is sent to the Global Aggregator, from the protection services that work when the data is downloaded, which must be set up after the data passes the Global Dispatcher (CA, decryption, key refreshment). The encrypted cloud storage is also separated. The whole ecosystem is shown in Figure 6.

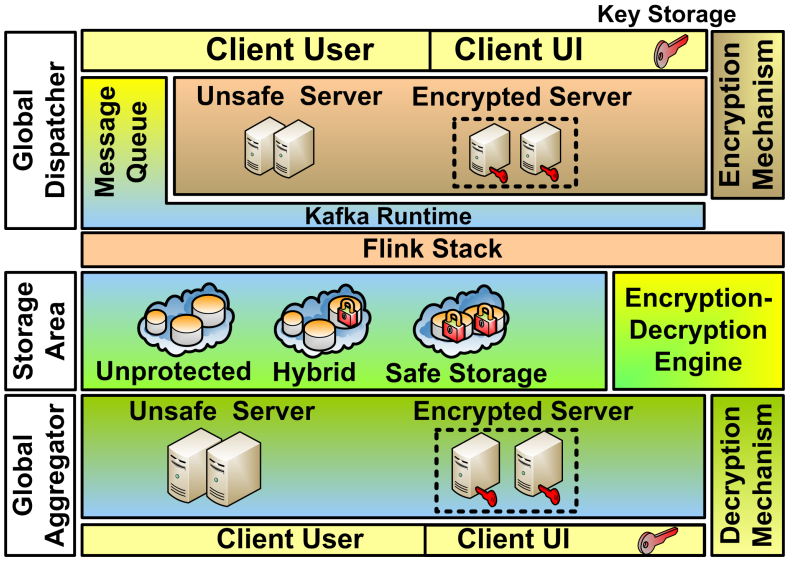

Figure 6. Components of the security infrastructure and their interactions

This integration model makes it possible to implement the required level of security and user privacy in all types of governmental and corporation organizations and services (like medical or governmental services) using the SMART infrastructure. The modular implementation design allows protecting the user privacy and control the access to the sensitive data in the heterogeneous environment via the Client UI developed for various OS and various platforms. The security is globally supervised with the help of the Encryption server and separate key storage. Additionally, this architecture can be easily extended in order to use more sophisticated methods of ensuring the data and key protection, *i.e.*, secret sharing or complex attribute policies.

5. Implementation and Results

The proposed Fast-Sec security infrastructure is currently implemented as part of the Storgrid project [32]. Storgrid is a secure cloud storage management system which allows controlling the level of data security and user privacy for a particular organization. The basic security components, *i.e.*, server UI, client UI and protected file storage have successfully passed the testing phase. The results and analysis from this testing phase are here presented. The tests are focused in evaluating the efficiency in attributing security levels to protected and public files. The administrator (or domain manager) controls this process in the server accessing it through a secret password. His/her responsibility is to decide the level of security of the protected/public files and define an authorization policy on a web interface shown in Figure 7.

The basic encryption method on the server side is 128 bit AES, which is a well-tested method that provides a fast encryption of the big bulks of data [20]. In order to enhance

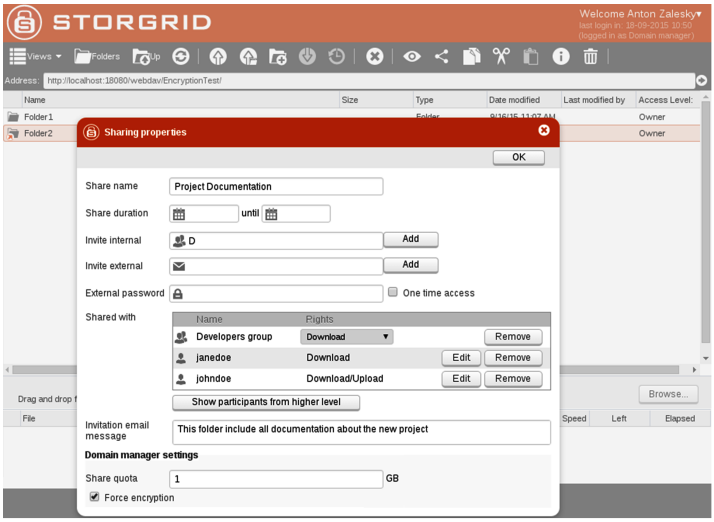

Figure 7. Storgrid web UI

the security, the file storage secret key has expiry period (e.g. one day), which forces a new re-encryption of the whole storage. However, in order to not negatively affect the system performance, the procedure of key regeneration and re-encryption runs when the system is idle or has low activity. The key expiry period makes the whole system more secure, due to the fact that hackers can hardly steal the key and the encrypted files within one attack. Moreover, auditing and log analysis performed on the server side do not allow a successful attack within one day.

The results of the re-encryption tests taken from the server real-time log in the above described scenario are shown in Table 1. The acquired results demonstrate that the average encryption time is rather low, even for the big bulk of data, considering the size of the data collected by the SMART services. The encryption time depends on the server load at the time when encryption has been performed. The results demonstrate a low overhead to the protected file re-encryption in local file system. Therefore, this re-encryption mechanism is suitable to be successfully integrated into the existing SMART infrastructure with the purpose of enhancing the security of Big Data storage.

Another reason to use re-encryption is that *Fast-Sec* does not use the concept of complete data encryption. Instead, it performs a selective encryption of the sensitive user data in order to avoid overhead that would occur if the encryption was performed over all the data. To perform this selective encryption, a given number of files, or fields in the database, are not encrypted while others are encrypted and sent over the insecure network (in the cloud). This concept avoids network overload and allows controlling the security of the files. The control of this process is performed by the domain manager access control utility.

Table 1. Re-encryption time results

Server Log	Files Number	Duration (ms)
srv.log_2015-09-18 18:06:28; [id = 1003;]	16,304	324.345
srv.log_2015-09-18 18:06:28; [id = 1004;]	35,834	162.629
srv.log_2015-09-18 18:07:23; [id = 1005;]	21,501	124.399
srv.log_2015-09-18 18:12:32; [id = 1006;]	23,651	149.948
srv.log_2015-09-18 18:40:00; [id = 1007;]	28,614	523.232
srv.log_2015-09-18 18:40:00; [id = 1008;]	49,494	397.334
srv.log_2015-09-18 18:55:02; [id = 1008;]	41,751	533.838
srv.log_2015-09-18 19:01:38; [id = 1010;]	45,360	368.900

By its turn, the client controls security from the user side and in the cloud. This control is implemented on different platforms, according to the specific control that is under

concern. The proposal supports the concept of light-weighted client, *i.e.*, it the client side is not responsible for executing much encryption or other critical/heavy operations. All encryption-heavy operations are performed on the server side, while the client just supports the key usage.

It is important to highlight that a unique feature provided by the proposed Fast-Sec is that when the encrypted files are not downloaded again, the keys remain the same. This allows the user manipulate his/her downloaded files after authentication with the password and user ID (his/her email), and the files are kept securely in his/her storage device (encrypted with the unique AES key). The access to the AES key is controlled by the server, following an ABE encryption and the above explained key expiry period. The operation of regenerating the public keys and resending them to users are not time-consuming due to the implementation design. This statement is supported by the results of tests in which the user key generation procedure for 1000 user logins and 1000 key samples out of 20 parallel threads, which are presented in Table 2.

Table 2. User key generation procedure results

User numbers	Average	Min	Max	KB/s	Avg. Bytes
30	0.82	38.4	35.7	154	320
60	0.67	42.1	34.7	138	340
120	0.52	45.1	34.0	124	370

Analyzing the results presented above, it is possible to state that the proposed Fast-Sec security infrastructure is light-weighted and designed to reduce the network load without compromising the level of user security. These are desirable properties for cloud-based systems that have to process huge amounts of data of different users and that possibly need to meet (near) real time requirements. Observing the context of SMART architecture and its intended usage, discussed in 1, the proposed Fast-Sec approach perfectly matches it's security needs without incurring in prohibitive overhead.

6. Conclusion

This paper reports the proposal of Fast-Sec, a security infrastructure solution for Cloud-based Big Data Analysis systems. The goal of the proposal is to address the security needs of cloud-based systems that handle massive amounts of data in real time analysis provided by heterogeneous data sources. This context is presented within a cloud-based architecture called SMART, which presents all featured characteristics of such emerging cloud-based systems. *Fast-sec* provides authentication and access control mechanisms appropriate to deal with the high demands of the applications intended to be supported by SMART architecture. The proposed infrastructure was designed and implemented to meet strict performance requirements, avoiding overloading the system with prohibitive overhead due to its security mechanisms. The performed tests provide evidence that the integration of the security infrastructure into the SMART architecture imposes low overhead without compromising the security level of the handled data, thus presenting itself as suitable for SMART and other similar cloud-based architectures.

Future works are planned to fully integrate the proposed security infrastructure in the SMART architecture. Additional load tests are to be performed in order to assess its scalability under a variety of system usage profiles, *i.e.*, under different demands of the real time Big Data analysis and varying types of data sources, besides those already tested and reported so far. Moreover, additional mechanisms can be incorporated to the infrastructure, such as data anonymization, which is very useful for different emerging cloud-based applications.

References

[1] D.A. Pereira, W.O. Moraisde , and E.P. Freitasde , *Nosql real-time database performance comparison*, International Journal of Parallel, Emergent and Distributed Systems 0 (2017), pp. 1–13, URL http://dx.doi.org/10.1080/17445760.2017.1307367.

[2] J. Choi, D.L. Nazareth, and T.L. Ngo-Ye, *The effect of innovation characteristics on cloud computing diffusion*, Journal of Computer Information Systems 0 (2017), pp. 1–9, URL http://dx.doi.org/10.1080/08874417.2016.1261377.

[3] W. Jansen and T. Grance, *Guidelines on Security and Privacy in Public Cloud Computing*, Tech. Rep. 800-144, 2011, URL http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf, Accessed in September 2015.

[4] M. Stonebraker, S. Madden, and P. Dubey, *Intel "Big Data" Science and Technology Center Vision and Execution Plan*, SIGMOD Rec. 42 (2013), pp. 44–49, URL http://doi.acm.org/10.1145/2481528.2481537.

[5] J.C.S. Anjos, G. Fedak, and C.F.R. Geyer, *BIGhybrid: a simulator for MapReduce applications in hybrid distributed infrastructures validated with the Grid5000 experimental platform*, Concurrency and Computation: Practice and Experience 1 (2015), pp. 1–24, URL http://dx.doi.org/10.1002/cpe.3665, Cpe.3665.

[6] G. Xu, W. Yu, Z. Chen, H. Zhang, P. Moulema, X. Fu, and C. Lu, *A cloud computing based system for cyber security management*, International Journal of Parallel, Emergent and Distributed Systems 30 (2014), pp. 29–45, URL http://dx.doi.org/10.1080/17445760.2014.925110.

[7] M. Ahmed and A.T. Litchfield, *Taxonomy for identification of security issues in cloud computing environments*, Journal of Computer Information Systems 0 (2016), pp. 1–10, URL http://dx.doi.org/10.1080/08874417.2016.1192520.

[8] V.C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, *Computer Security*, Tech. Rep. 800-162, 2014, URL http://dx.doi.org/10.6028/NIST.SP.800-162, Accessed in September 2015.

[9] J. Whitworth and S. Suthaharan, *Security Problems and Challenges in a Machine Learning-based Hybrid Big Data Processing Network Systems*, SIGMETRICS Perform. Eval. Rev. 41 (2014), pp. 82–85, URL http://doi.acm.org/10.1145/2627534.2627560.

[10] Q. Liu, C. C.Tan, J. Wu, and G. Wang, *Reliable Re-Encryption in Unreliable Clouds*, in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec, 2011, pp. 1–5.

[11] D. Chen and H. Zhao, *Data Security and Privacy Protection Issues in Cloud Computing*, in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, Vol. 1, March, 2012, pp. 647–651.

[12] S. Rajan, W.V. Ginkel, N. Sundaresan, *et al.*, *Expanded Top Ten Big Data Security and Privacy Challenges*, Tech. rep., 2013, URL https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf.

[13] J.C.S. Anjos, M.D. Assuncao, J. Bez, C.F.R. Geyer, E.P. Freitasde , A. Carissimi, J.P.C.L. Costa, G. Fedak, F. Freitag, V. Markl, P. Fergus, and R. Pereira, *SMART: An Application Framework for Real Time Big Data Analysis on Heterogeneous Cloud Environments*, in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*, Oct, Liverpool, UK, 2015, pp. 199–206.

[14] W. Zhang, Y. Lin, S. Xiao, Q. Liu, and T. Zhou, *Secure distributed keyword search in multiple clouds*, in *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, May, 2014, pp. 370–379.

[15] C. Zhang, E.C. Chang, and R.H. Yap, *Tagged-MapReduce: A General Framework for Secure Computing with Mixed-Sensitivity Data on Hybrid Clouds*, in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May, Chicago, IL, USA, 2014, pp. 31–40.

[16] V.C. Hu, T. Grance, D.F. Ferraiolo, and D.R. Kuhn, *An Access Control scheme for Big Data processing*, in *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, Oct, Miami, Florida, USA, IEEE Computer Society, 2014, pp. 1–7.

[17] B. Lublinsky, K.T. Smith, and A. Yakubovich, *Professional Hadoop Solutions*, 1st ed., John Wiley & Sons, Inc., 10475 Crosspoint Boulevard, 2013, p. 504.

[18] K. Zheng and W. Jiang, *A token authentication solution for hadoop based on kerberos pre-authentication*, in *Data Science and Advanced Analytics (DSAA), 2014 International Conference on*, Oct, 2014, pp. 354–360.

[19] T. Galibus and H. Vissia, *Cloud storage security*, in *Network Security and Communication Engineering*, K. Chan, ed., CRC Press, 2015, pp. 123–126.

[20] U.S. Commerceof , *Announcing the Adcanced Encryption Standard (AES)* , Tech. rep., (NTIS), 5285 Port Royal Road, Springfield, VA 22161, 2001, URL `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`, Accessed in September 2015.

[21] V.C. HU, D.R. KUHN, T. XIE, and J. HWANG, *Model Checking for Verification of Mandatory Access Control Models and Properties*, International Journal of Software Engineering and Knowledge Engineering 21 (2011), pp. 103–127.

[22] W. Zeng, Y. Yang, and B. Luo, *Access control for big data using data content*, in *Big Data, 2013 IEEE International Conference on*, Oct, Silicon Valley, CA, IEEE Computer Society, 2013, pp. 45–47.

[23] J. Bethencourt, A. Sahai, and B. Waters, *Ciphertext-Policy Attribute-Based Encryption*, in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, May, 2007, pp. 321–334.

[24] V. Goyal, O. Pandey, A. Sahai, and B. Waters, *Attribute-based Encryption for Fine-grained Access Control of Encrypted Data*, in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, Alexandria, Virginia, USA, URL `http://doi.acm.org/10.1145/1180405.1180418`, ACM, New York, NY, USA, 2006, pp. 89–98.

[25] Z. Qiao, S. Liang, S. Davis, and H. Jiang, *Survey of attribute based encryption*, in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*, June, 2014, pp. 1–6.

[26] Z. Liu and D.S. Wong, *Practical Attribute-Based Encryption: Traitor Tracing, Revocation, and Large Universe* (2014), Http://eprint.iacr.org, Cryptology ePrint Archive, Report 2014/616.

[27] D.H. Le, H.L. Truong, G. Copil, S. Nastic, and S. Dustdar, *SALSA: A Framework for Dynamic Configuration of Cloud Services*, in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, Dec, 2014, pp. 146–153.

[28] S. Delamare, G. Fedak, D. Kondo, and O. Lodygensky, *SpeQuloS: a QoS service for BoT applications using best effort distributed computing infrastructures*, in *Proceedings of the 21th international symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, Delft, The Netherlands, URL `http://doi.acm.org/10.1145/2287076.2287106`, ACM, New York, NY, USA, 2012,

pp. 173–186.

[29] T. Zhang, *Reliable Event Messaging in Big Data Enterprises: Looking for the Balance Between Producers and Consumers*, in *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems*, DEBS '15, Oslo, Norway, ACM, New York, NY, USA, 2015, pp. 226–233.

[30] A. Alexandrov, R. Bergmann, S. Ewen, J. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M.J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, *The Stratosphere platform for big data analytics*, VLBD Journal 23 (2014), pp. 939–964.

[31] L. Mashayekhy, M. Nejad, and D. Grosu, *A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources*, Parallel and Distributed Systems, IEEE Transactions on PP (2014), pp. 1–14.

[32] B.D.S. BV, *Storgrid EFSS: Secure Enterprise File Sharing Software* (2016), Available from Internet: http://www.storgrid.com/.