# A simple architecture for secure and private data sharing solutions

Antonio Famulari, Francesco Longo, Giuseppe Campobello, Thomas Bonald, Marco Scarpa

# A Simple Architecture for Secure and Private Data Sharing Solutions

Antonio Famulari*, Francesco Longo†, Giuseppe Campobello†, Thomas Bonald* and Marco Scarpa†
* Tèlècom ParisTech, INFRES, France
{famulari,bonald}@telecom-paristech.fr
† Università degli Studi di Messina, Italy
{flongo,gcampobello,mscarpa}@unime.it

***Abstract*—In recent years, Storage as a Service Cloud gained popularity among both companies and private users. However, security and interoperability issues still have to be adequately faced and solved. In this work, we propose a simple, secure, and privacy-preserving architecture for inter-Cloud data sharing. The proposed solution relies on open standards for both sharing and communication mechanisms, thus ensuring durability, robustness, and compatibility of the approach in the current Internet.**

***Keywords*—Storage Cloud, Security, Interoperability, Data Sharing**

## I. Introduction

Cloud computing is a challenging technology that promises to strongly modify the way computing and storage resources will be accessed in the near future. The traditional taxonomy about Cloud distinguishes among services at three different levels: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The above reported services can be implemented in both private and public Clouds. While private Clouds have infrastructures owned and maintained by a single organization for its own use, public Clouds are made available to the general public by a service provider.

Recently Storage as a service (STaaS) Cloud gained popularity both among private users and enterprises [1]. STaaS is a Cloud business model in which a service provider rents space in its storage infrastructure to individuals or companies. In this context, some of the main challenges, both from an industry and an academy point of view, are related to security (particularly associated to the personal sphere of users, compliance with legislation, and problems of trust) and interoperability.

Security problems [2] are connected to the value and sensitivity of the data stored in the Cloud. On the one hand, big industrial groups might hesitate to put their sensitive data in the Cloud, under the control of other organizations, if any guarantee of privacy, integrity, and availability of data is not provided. Besides data themselves, privacy of related metadata and interactions is also important. In fact, such information is usually fully accessible to the Cloud operator. On the other hand, common users do not usually care about privacy and security of their data. Not subjected to strong regulation as industrial groups or unaware of risks and implications for their privacy, they tend to accept terms of use and conditions imposed by the different Cloud operators.Nevertheless, problems for user privacy are still an actual menace.

Besides privacy and security concerns, with respect to big industrial groups, common users are more concerned with the other challenge of Cloud computing, that is interoperability [3]. In fact, several Cloud services cannot interoperate among each other due to the absence of standard mechanisms. As a consequence, in order to fully profit of interaction facilities, two generic users usually need to rely on the same STaaS provider. This also makes difficult to migrate data from one platform to another without relying on local equipment.

Several works in literature deal with security [4] [5] [6] and interoperability [3] [7] [8] issues focusing on standardization of Cloud services. Our goal is not to provide a new standard but to propose an approach that is transparent for both users and providers. In particular, we present a solution that is able to allow integration among different STaaSs without asking for changes in their internal policies or implementations. As a consequence, service providers can be unaware of the solution. Note that our work is not focused in presenting new encryption mechanisms or techniques. In fact, our solution is more targeted on privacy preservation and with this aim it is able to directly exploit a variety of encryption algorithms already present in literature with minor changes in the design and implementation. Our proposal relies on: i) a user client containing all the added complexity, ii) out of band (OOB) communication mechanisms (e.g., email or XMPP services) used for signaling purposes, and iii) potentially any existing STaaS with minimum sharing requirements for user data storing.

## II. Motivations and requirements

Two kind of requirements have been identified: privacy-related requirements and functional requirements.

### A. Privacy-related requirements

In the context of the present work, privacy is intended as the effectiveness for a user to be able to restrict the access to data he/she is responsible for (e.g., as a producer of the content) or related to (e.g., details about his/her interactions). We thus differentiate among privacy of data and privacy of metadata. Such an effectiveness can be affected by technical and/or social (e.g., social engineering) interferences [9].

**Privacy of data.** Privacy of data is satisfied if a user, as producer or owner of a given piece of content, is able to define who can access it and he/she is not obliged, by construction of the platform, to make it accessible to any other party, included any service provider or any other entity running the service. We believe that not only the user should have an exact perception

of who can access his/her data, but that it should be possible to assure that nobody but intended parties may access shared content.

**Privacy of metadata.** Metadata attached to user data may sometimes reveal more information than the data itself. As an example, let us consider metadata of a photograph: it may contain several sensible information such as GPS position, date and time of the photograph, device that has been used, and so on. We believe that medatada should be protected neither more nor less than data, and it should be exclusively accessible to intended parties [10]. Moreover, nobody but interacting parties should be aware of any social interaction/relationship.

### B. Functional requirements

In the present work, we focus on functional requirements from two main point of views. On the one hand, we put our attention to the flexibility in the choice of the external resources and services that user can exploit. On the other hand, the granularity of the access control mechanisms is taken into consideration.

**Flexible choice of external resources.** One of the main functionalities that users ask for when using STaaS platforms is the possibility to share content with other people. However, for several reasons (e.g., cost, trust toward the provider), users often rely on different storage services [6]. In this case, the interaction capabilities are rather limited to the simple link sharing, and rely OOB (with respect to the exploited platforms) mechanisms such as e-mail or instantaneous messages (IM). Also, to the best of our knowledge, data migration from one platform to another is not supported by any standard mechanism. We believe that an effective sharing mechanism should guarantee full interoperability among different STaaS platforms, allowing users to interact also with principals using different services, to easily define sharing groups, and to migrate from one service to another in a simple way without relying on local equipment.

**Access control.** Typical access control mechanisms are much harder to be implemented in a decentralized platform with respect to centralized counterparts. However we believe that also in a decentralized sharing platform, a user should be able to define, in a simple way, with whom to share a piece of content and to easily assign and revoke such privileges.

## III. Architecture overview and functionalities

In this section, we provide an overview of the proposed architecture and we describe the main functionalities.

### A. Architecture entities

The architecture we propose in the present work is composed of three main classes of entities: (i) one or more STaaS spaces which support easy sharing mechanisms, such as HTTP or WEBDAV, used for storing data, metadata, and any information about user interactions; (ii) one or more communication services, such as e-mail or IM services, used for the signaling mechanism; (iii) one local client on each device of each user.

**STaaS spaces.** Our architecture is designed in order to work with several heterogeneous STaaS spaces. The configuration process consists of setting a vector <URL, credentials> for each of the STaaS spaces that the user is willing to use to store his/her data. The credentials will be provided to the client so that it will be able to login and access the storage spaces. Each user will *only* write on his/her storage space, which we will refer at as *personal space*.

**Communication services.** Any e-mail or IM service can be used for the signaling mechanism. In general, users should reserve one or more accounts for the platform, so that signaling messages related to our sharing mechanism do not interfere with common messages.

**User client.** The local client is the core of the system and provides a set of primitives/functionalities that can be used to implement a rich interaction model relying on the STaaS spaces for storing data, metadata, and information about interactions and on the communication services for the signaling and interactions. The interaction model, in terms of functionalities is independent from the used STaaS and may also be exploited by third party applications which can be implemented on the top of our client.

### B. Configuration

The user needs to install the client on each local equipment that he/she wants to use and configure one or more storage spaces. Also, he has to configure one or more e-mail or IM accounts, which should be reserved for the platform usage. Then, he/she has the possibility to add a certain number of *colleagues*. Each colleague is identified by an identifier (e.g., name, nickname, email address). To such identifier the following items will be linked: (i) one or more URLs of the personal space of the colleague which will point to a file, called *activity stream* where the colleague will report his activities, and to an entry point to the colleague's data; (ii) a security association between the user and the colleague, e.g., a key or a set of keys, so that the user and the colleague may communicate and share content in a secure way; (iii) optionally, an e-mail or IM contact reserved to such usage that will be used for signalling purposes.

### C. Working principle

The proposed solution implements primitives for storing content and sharing with other colleagues.

**Store a content.** Each user authenticates and stores data *only* on his/her own personal spaces, hosted by a STaaS on his choice. STaaS authentication vectors will never been shared with other users. Data are always stored in an encrypted form, using encryption and key handling mechanisms embedded in the client. Contents that are not intended to be shared are stored in an encrypted way with keys that are known only by the owner of the content.

**Share a content.** Stored content can thus be shared via public sharing links, so that anybody can download it. However, in order to access a content a user has to own the decryption keys. The application will handle encryption/decryption operations as well as assignment and revocation of privileges (in the form of keys for accessing content) and notification.

In particular, when a user shares a piece of content with a colleague, the content is encrypted with a fresh symmetric key, which in turn is shared with the recipient of the content in a secure way by mean of asymmetric public-key cryptography. Similarly, when a colleague shares a piece of content with groups of users, the piece of content is encrypted with a fresh symmetric key and only one copy of the encrypted content is produced. The symmetric encryption key is thus encrypted for the group using a group key, which is owned by all the members of a group, and shared among recipients. This assures that only intended parties may access the content. The recipient is then notified via mechanisms described below.

### D. Notification mechanisms

The notification mechanisms are one of the most important aspects of the proposed architecture. They must be able to provide easy inter-Cloud sharing, meaning that they do not have to rely on sharing facilities provided by specific storage providers. In fact, thanks to our architecture two generic users do not need to rely on the same providers for interaction. The architecture supports a proactive and a reactive notification mechanism.

**Proactive notification.** The first mechanism relies on e-mail or IM services to send notification messages to the colleague (or list of colleagues) to be notified. When a user chooses to share an information with someone else, a message is sent by either e-mail or IM mechanism with the link to that information into the STaaS provider infrastructure. Please note that the link to the content itself, embedded into the message, gives no access to the content, since the latter is stored in an encrypted form. The advantage of the proactive method is that thanks to "push" techniques, supported by IM and most e-mail services, an asynchronous notification is built.

**Reactive notification.** The second mechanism is based on a regular verification, performed by each user, of a *stream file* (namely Activity Stream) which is accessible on the public space of each colleague. According to such a mechanism, in order to notify a colleague of a new sharing, a user has to report such interaction in his/her activity stream. At the time of execution and regularly at predefined intervals, each client downloads and verifies the content of the activity streams of each or part of his contacts.

Note that the activity stream only contains entries addressed to specific contacts or groups of contacts. Specific entries are cryptographically protected for the user or the group of user, based on discussed security associations.

## IV. Implementation details

### A. User Client implementation

Figure 1 shows a graphical representation of the client layered architecture and its interactions with external STaaS spaces and communication services. Five modules are present in the User Client: User Interface/API, Social Engine, Encryption Module, Read/Write Module, and Signaling Module. The User Interface allows the user to interact with the system. The same layer also provides APIs to possible third party applications which can implement complex interactions based on basic functionalities provided by the system. The

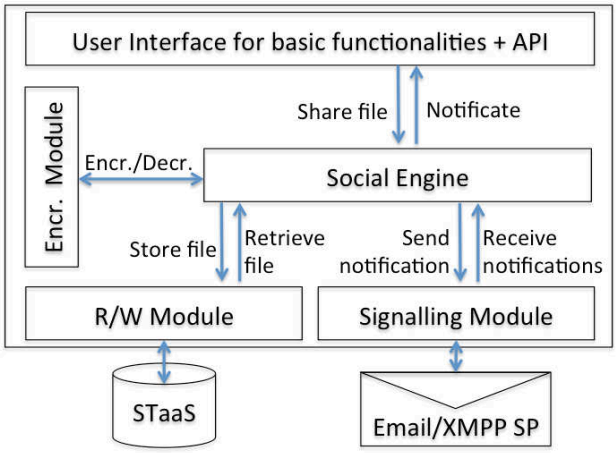


Fig. 1. Graphical representation of the proposed architecture.

Social Engine handles the system life cycle implementing the orchestration of the main functionalities. The Encryption Module allows to encrypt/decrypt pieces of content through the considered encryption method. The R/W Module behaves as an abstraction layer for the different STaaS spaces that the system can deal with. Such module will be able to write only on the user personal space and read from any storage. Finally, the Signaling Module implements the communication layer by abstracting the different communication service providers.

Two main high-level functionalities are present in our prototype (refer to Figure 2 for the steps described below). On one hand, the user is allowed to share a piece of data with other users by selecting it from the local file system. First of all, the user can indicate the identity of the users or groups of users with which he/she wants to share the data (1). Then, the encryption module is instructed for encrypting the piece of content accordingly (2). Afterwards, the encrypted piece of content is uploaded to the selected Storage Cloud through the Storage Module (3, 4). Finally, the corresponding notification is sent to such users through the selected signaling method by mean of the Signaling Module (5, 6).

On the other hand, the recipient client can receive signals about content that has been shared with him/her (8, 9). When a notification is received through the Notification Module, the Social Engine retrieves the content on the remote Storage Cloud on which it is stored through the Storage Module (10, 11). Then, the piece of data is decrypted by mean of the Encryption Module (12) and it is made available to the user through the User Interface (13).

### B. Adopted standards and technologies

We implemented a prototype of the proposed architecture on an Android Samsung Galaxy Note device. Dropbox and Google Drive have been selected as STaaS spaces while Gmail as been taken into consideration as communication service. Particularly interesting are the technologies adopted in order to implement reactive notification mechanisms and group encryption mechanisms.

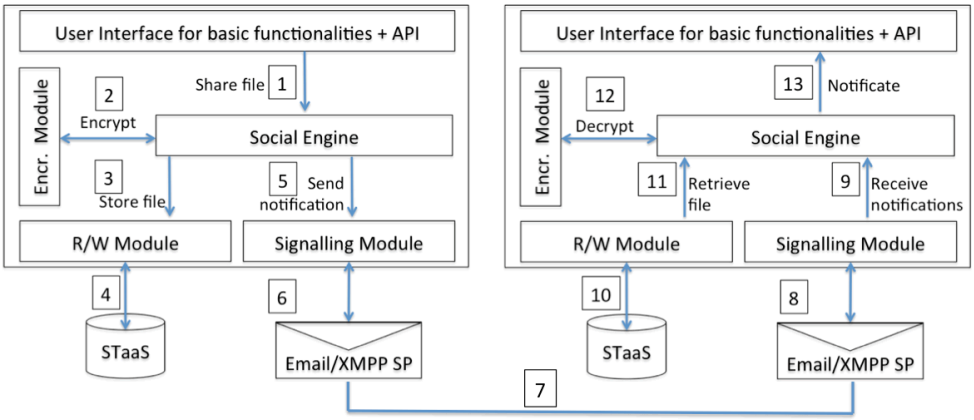


Fig. 2. Interactions between two clients.

**Activity description.** In order to implement reactive notification mechanisms we exploit the concept of Activity Stream. An Activity Stream is a list of recent activities performed by an individual in social web applications and services. The more known example of Activity Stream is Facebook's News Feed. Since the proliferation of the Activity Stream concept on websites, in recent year the necessity to standardize the format came up in order for different websites to interact among each other through their streams. In our work, we took into consideration the open format Activity Streams [11] and we designed a draft implementation of the format from both the producer and the consumer point of view.

**Group keying mechanisms.** So as to support easy sharing with locally defined groups of users, each principal may locally define groups of users such as friends, family, co-workers. Our proposal relies on Group Encryption with hierarchy of keys [4], which allows to define by construction of the group several subgroups. Such approach allows the issuer of the group to address content to the whole group as well as to one or more subgroups. It also allows for easy revoking privileges to one user, i.e. removing one user from a group, reducing the number and the complexity of the operations for re-keying remaining users.

```
{
"url": <URL>,
"author": <username>,
"published": "yyyy-mm-ddThh:mm:ssZ",
"updated":"yyyy-mm-ddThh:mm:ssZ",

"totalItems":45,
"itemsPerPage":15,
"startIndex":1,
"next": <next_stream_URL>,
"prev": <previous_stream_URL>,
"first": <first_stream_URL>,

        "main_activity_array":
                [
                        {
                         "published":"yyyy-mm-ddThh:mm:ssZ",
                         "id": <sequential_ID>,
                         "url": <List_ActivityStream_URL>
                         "target":"id-list"
                                 },
                ]
}
```

Fig. 3. Structure of the Main Activity Stream

### C. Reactive notification module

We adopted a JSON format for the activity streams. More in particular, a customization of the approach proposed by [11] has been implemented. In Figure 3, the structure of the Main Activity Stream is shown. Field "`url`" contains the URL of the user personal space, which is the entry point to his/her data while "`totalitems`" is the maximum number of objects that the stream may contain. This has been set up to 45 in order to maintain the activity streams small. As soon as the number of elements exceeds such limit, a new Activity Stream is generated, which links to the previous one via field "`prev`". Each Activity Stream also links to the following one via field "`next`". The current Activity Stream is always available at the same URL in order to facilitate the retrieval while previous Activity Streams are moved at different URLs as specified by fields "`prev`" and "`next`". Field "`first`" links to the first Activity Stream so that it is possible to go through the whole list of streams starting from the beginning. Field "`main activity array`" lists all the activities reported in the current stream. As discussed above, each activity reported in the Main Activity Stream is referred to a list of users and thus described in the related List Activity Stream, linked through field "`url`" of "`main activity array`".

List Activity Streams have a very similar structure with the exception that several objects link at the content rather than at other streams as in the case of Main Activity Streams. In Figure 4, the structure of object "`image`" in a List Activity Stream is depicted. Field "`title`" is a human readable message that is displayed to the user. Field "`url`" contains the URL of the personal space of the user while "`id`" is composed of the list-id joined with a sequential number for the list. Main Activity Streams are downloaded in parallel and the analysis of each of them starts as soon as the download is completed. Based on the content of the latter, List Activity Streams are possibly downloaded and analyzed.

```
"object":{
                    "title": <human_readable_description>,
                    "url": <userPersonalSpace_URL>,
                    "id": <list-id>_<counter>",
                    "image": <content_URL>
                            },
                },
```

Fig. 4. Structure of object "`image`" in a List Activity Stream

### D. Proactive notification module

We relied on email as messaging system. Messages follow JSON format very similar to the one adopted for Streams. The adopted email address is exclusively used for the system. As soon as a new activity is performed by the user, the latter can choose one or more recipients. Thus, the system automatically generates email messages addressed to the recipients with a description of the new activity. At the reception of the email, the client automatically reacts decrypting and analyzing the message and displaying the notification on the terminal.

### E. Internal database

The internal status of the application is maintained within an internal database which has been implemented in SQLite. Such database is stored in an encrypted form on the personal space of the user and contains basic information about the user (his email, personal information and keys), his contacts (e.g. associations between users and related keys, lists of friends and related keys) and his activities within the platform (for example, each sharing activity is reported). Such database is upgraded on the personal space of the user each time the latter launches the application, performs a significant action (e.g. adds a new contact, shares a new piece of content) and allows the user to accesses the platform from different devices. Based on our experience, the size of the SQLite database is few KB, and grows very slowly as the number of contacts or activities increases.

## V. Evaluation

In this section we provide both qualitative and quantitative evaluation.

### A. Qualitative Evaluation

The system we propose implements a simple co-working environment, allowing sharing of contents and providing a framework that can be exploited for building complex interactions, relying on several STaaS which are in general unable to interact.

More in details, qualitative advantages of our system are the following.

- It add new mechanisms for inter-Cloud data sharing among different STaaS without the necessity for them to change anything in their way of working, while previous works mostly focuses on standardization of Cloud to Cloud interfaces [12], [13]. Complex functionalities can be implemented as plugins on the top of the system, which provides primitives for interactions.
- It only relies on basic services of the Internet for sharing content (e.g., HTTP, WebDAV) and communication (e.g., e-mail, IM). Relying on these services ensures durability, robustness, and compatibility of the service in the current Internet.
- It assures good levels of protection for the personal sphere of users: all data stay under the control of the owners and remain inaccessible to any unauthorized party, included the storage service. Besides, access to data is as anonymous as possible, since the access control relies on encryption and users never have to authenticate in order to access a piece of content. The Cloud provider can only track accesses based on the IP addresses, but this kind of tracking has always been there since the born of Internet. Also, several project aiming at providing anonymity at IP level already exist, such as the TOR project, which allow to avoid IP-based tracking.
- It abstracts from the exploited storage service: each user may rely on the service of his choice, provided it allows to share a piece of content via a sharing link. Each user will always write on his/her own space. Each sharing, modification, comment to a content can only be done by uploading content on the own storage space and relying on the interaction mechanism for notification.

### B. Quantitative Evaluation

We consider the dependency on the signalling module, which for some interactions is based on the publish/subscribe approach, as critical with respect to system performance. In fact, in order to display the list of activities of a user contacts it is necessary to download, decrypt, and analyze several activity streams.

With the aim of evaluating the efficiency of such approach, we tested our prototype in the more demanding case of social network interaction dynamics, since inter-cloud sharing and social network platforms are collapsing in one unique interaction model. Also, although inter-cloud sharing systems are mostly intended for usage from a PC, we also considered usage from smart phones. We thus tested the publish/subscribe approach on both a mobile device (Android Samsung Galaxy Note) and a PC (Pentium 4, 2GHz, 1GB RAM) and observed a very weak dependence on the local hardware with respect to other parameters. More in particular, performance depends on several variables: (i) Internet connection used for downloading user activity streams; (ii) performance of the different services users leverage for storing their own data; (iii) number of contacts of which it is necessary to download, decrypt, and analyze the related activity streams; (iv) number of activities of such contacts in the context of the social network; (v) employed device.

For what concerns the Internet connection, we tested the mechanism in the case of both the Ethernet-based Internet connection of our University Department and a 3G Internet connection (usually used on smart phones). As storage services we relied on Dropbox. Concerning the number of user contacts, we identified 3 different scenarios which are typical of social network platforms [14]: users with a small number of contacts

(25 contacts), users with average number of contacts (100) and users with a huge number of contacts (500). However, the majority of social network users tend to be interested at and to interact with a much smaller number of contacts [15], that is usually about 20% and however never more than 50 contacts; that is interactions with 5, 20 and 50 contacts in the three discussed cases. That means a social system can give higher priority to such subsets of contacts for providing a first activity stream of most important contacts, then download in the background data for all the remaining contacts. Note that this approach is somewhat similar, in terms of user experience, as what common social networks provide by allowing users to *favorite* some contacts for filtering out most interesting entries in the activity stream. For what concerns the activity of contacts, we considered average levels of activity for all contacts, that is about five activities per day [16]. However, among all the activities a user can perform, only a subset of them are notified via the publish/subscribe mechanism, that is notification to a wide range of contacts; while others, for example direct one-to-one interactions, are notified via direct messages (e.g., IM, email). Such distinction is not foregone, and depends on design choices. With respect to the list of typical activities the ones notified via the publish/subscribe model are: (i) new friendship/contact: since in general new relationships are notified to all contacts; (ii) posted a status update, a new picture, or a wall post etc.: since that concerns several contacts of a user; (iii) liked a friend's content: so as to implement a basic recommendation system.

Such subset of activities represents about 47% of all typical day's activities [16]. Results of the simulations for the three cases are summarized in Table 1. We report in parentheses the results obtained by considering the subset of maintained relationships only, as discussed above.

TABLE I. PERFORMANCE OF THE SYSTEM

| Connection | 25 (5) contacts | 100 (20) contacts | 500 (50) contacts |
|---|---|---|---|
| University | 4.6 (1.6) sec. | 5.6 (4.6) sec. | 15.5 (5.2) sec. |
| 3G | 10.9 (3.8) sec. | 34.5 (13.5) sec. | 48.5 (13.5) sec. |

The bottleneck seems to reside in the download of the several activity streams, while decryption and analysis operations seem to have a smaller impact on overall performance of the system. That means the system remains usable in the case of University Internet connection in all the cases except the most demanding one (a user with 500 contacts who is interested at the activities of all his contacts, which in general is not the case). On the other hand, that is with a 3G Internet connection, the system seems to be usable only in the case where one is interested at a very small subset of contacts. However, in the case of inter-cloud sharing users tend to maintain a very small number of contacts.

## VI. CONCLUSIONS AND FUTURE WORK

In this work, we proposed a new approach for inter-Cloud data sharing. We first identified the main requirements and then provided both high-level and implementation-level details of our approach. Relying on open standards, such as HTTP, WebDAV, e-mail, and IM, our approach ensures durability, robustness, and compatibility of the approach in the current Internet by providing security and interoperability. In the future, we aim at implementing a more complete prototype of our architecture and quantitatively investigating functional and non-functional requirements. Moreover, we plan of building on top of our architecture by implementing a distributed, shared, and secure file-system exploiting STaaS spaces available online.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the Italian National project "Integrated Cloud-Sensor System for Advanced Multirisk Management" under grant agreement PON01_00683.

## REFERENCES

[1] E. Kolodner, S. Tal, D. Kyriazis, D. Naor, M. Allalouf, L. Bonelli, P. Brand, A. Eckert, E. Elmroth, S. Gogouvitis, D. Harnik, F. Hernandez, M. Jaeger, E. Lakew, J. Lopez, M. Lorenz, A. Messina, A. Shulman-Peleg, R. Talyansky, A. Voulodimos, and Y. Wolfsthal, "A cloud environment for data-intensive storage services," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, 29 2011-dec. 1 2011, pp. 357 –366.

[2] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *Services Computing, IEEE Transactions on*, vol. 5, no. 2, pp. 220 –232, april-june 2012.

[3] S. Dowell, A. Barreto III, J. B. Michael, and M. T. Shing, "Cloud to cloud interoperability," in *Proc. of the 2011 6th International Conference on System of Systems Engineering*, Jun 2011, pp. 258 – 263.

[4] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16–30, 2000.

[5] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, march 2010, pp. 1 –9.

[6] J. Li, C. Jia, J. Li, and Z. Liu, "A novel framework for outsourcing and sharing searchable encrypted data on hybrid cloud," in *Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on*, sept. 2012, pp. 1 –7.

[7] S. Labes, J. Repschlager, R. Zarnekow, A. Stanik, and O. Kao, "Standardization approaches within cloud computing: Evaluation of infrastructure as a service architecture," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, September 2012, pp. 923 –930.

[8] S. N. I. Association and the Open Grid Forum, "Cloud storage for cloud computing," in *Open Grid Forum, Storage Networking Industry Association*, September 2009, pp. 1 –12.

[9] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and privacy in cloud computing: A survey," in *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, nov. 2010, pp. 105 –112.

[10] B. Snively, T. I. Morris, R. Ita, and K. L. Fox, "An application of security access and control to semantic metadata management," in *Military Communications Conference, 2007. MILCOM 2007. IEEE*, oct. 2007, pp. 1 –7.

[11] "The activity streams format6." [Online]. Available: http://activitystrea.ms/

[12] (2013) Cloud data management interface (CDMI). [Online]. Available: http://www.snia.org/cdmi

[13] X. Luo and H. Li, "Experiment design for cloud storage application based on cdmi," in *Open-Source Software for Scientific Computation (OSSC), 2011 International Workshop on*, 2011, pp. 148–152.

[14] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[15] (2013) Maintained relationships on facebook. [Online]. Available: https://www.facebook.com/note.php?note_id=55257228858&ref=mf

[16] (2013) Typical daily facebook activities. [Online]. Available: http://www.statista.com/statistics/192716/typical-daily-facebook-activities-of-us-users/