# *SeC*$^2$: SECURE MOBILE SOLUTION FOR DISTRIBUTED PUBLIC CLOUD STORAGES

Juraj Somorovsky[1], Christopher Meyer[1], Thang Tran[2], Mohamad Sbeiti[2], Jörg Schwenk[1] and Christian Wietfeld[2]

[1]*Horst Görtz Institute for IT-Security (HGI), Ruhr-University Bochum, Bochum, Germany*

[2]*Communication Networks Institute (CNI), Faculty of Electrical Engineering and Information Technology TU Dortmund University, Dortmund, Germany*

Keywords: Cloud Security, XML Encryption, Seamless Networking, Virtual Private Storage.

Abstract: Cloud Computing is an emerging technology that attracts increasing attention as a high-performance and low-cost solution to process arbitrarily huge data and share them among different users and organizations. Nonetheless, this technology rises new questions on security and privacy, especially when working with highly confidential data. Existing solutions offering end-to-end security such as TLS are of no help since the stored data is only encrypted during their transport. Therefore, a message-level security must at least be applied to protect those data during and after the storing process. This paper investigates a novel solution for secure data storage in the cloud. It presents a security concept allowing each client to encrypt outgoing data on one's mobile device and share it among a defined user group while using a seamless service provision. As our concept is used transparently as well as independently on each device, users keep full control over their data and almost no changes on the existing infrastructure of cloud storage services are needed.

## 1 INTRODUCTION

Cloud Computing has become a well-used paradigm in the recent years as reported by Zhang et al. (Zhang et al., 2010). It brings advantages to both clients and services. The emerging technology allows users for private and business purposes to access all their applications and documents at anytime and anywhere using different mobile devices (e.g. laptops or smart phones). Companies, on the other hand, do not need to spend a great amount of time and money on the management of their IT-infrastructure for application processing and data storage anymore. Using cloud computing technology, the time spent on lower-value activities can be minimized and thereby companies can focus more on strategic activities with greater impact on their business. These considerations were provided by Molnar and Schechter (Molnar and Schechter, 2010).

To increase the user acceptance of cloud services, the fulfillment of different requirements concurrently, such as security and high availability in a heterogeneous network, is still a challenge. This has also been observed by Wu et al. (Wu et al., 2010). In the past, data could be secured by means of, e.g., personal firewalls or assignment of permissions. However, with increasing mobility of end-user devices as well as services, the requirement for storing data has changed these days. Due to this trend the outsourcing of data to cloud storages is always fraught with uncertainty as users have absolutely no control over their data. This, in particular, represents a huge problem regarding storage of confidential information. Encryption services which allow to encrypt users' confidential data on their servers (e.g. Dropbox[1]) are of no help. Although they allow a secure storage of users' data in the cloud, the clients must still build trust relationships with the encryption services provided by the cloud (proper encryption on server side, no backup keys, backdoor-free systems, no data disclosure to third parties, ...). Therefore, the security processing must be moved to the cloud client in order to give him the possibility to take full control over his data.

Besides the security aspects, today's mobile communication is available anywhere where service providers have to ensure a high availability of their

[1]http://www.dropbox.com

Somorovsky J., Meyer C., Tran T., Sbeiti M., Schwenk J. and Wietfeld C..
SeC2: SECURE MOBILE SOLUTION FOR DISTRIBUTED PUBLIC CLOUD STORAGES.
DOI: 10.5220/0003903205550561
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 555-561
ISBN: 978-989-8565-05-1

provided cloud services. On the one hand, this requires reliability solutions for the cloud services in the back-end as described by Wang et al. (Wang et al., 2010). On the other hand, as reported by Tran et al. (Tran et al., 2010), a deployable and capable vertical handoff (VHO) solution for mobile devices is needed in the front-end. The use of VHO allows mobile clients seamless access to services across different mobile access technologies.

The main contribution of this paper is the proposal of a cost-efficient and feasible security framework for mobile devices that provides a holistic security solution with full data control for users. These security goals are achieved by a so called *Sec*$^2$ middleware that encrypts and decrypts the communication between mobile devices and cloud servers. The *Sec*$^2$ middleware is based on the XML Encryption standard (Eastlake et al., 2002) and allows the applications to securely store XML data in the cloud as for example on XML databases. This gives the clients the possibility of a seamless and secure storage solution for their data without establishing trust relationships with cloud services.

We present a system of group keys, which allows to create different user groups and authorize them for decryption of dedicated parts of documents.

Besides the security capabilities, our framework includes a mobility support that allows seamless service provisioning in a heterogeneous network.

## 2 RELATED WORK

The topic *Storage Services* is not new and is already offered by several technologies like Microsoft Sharepoint[2], OpenText[3] and Dropbox that are mostly accessible by web interfaces. Except for the occasional one, e.g., Brainloop[4], OceanStore[5] and Dropbox, current systems support cryptographic protection for stored data, which is, however, saved on untrusted cloud servers. Moreover, these approaches focus on some kind of trust relationship between the mobile client and the cloud service. Clients have to trust the service provider not to disclose stored data and properly secure its infrastructure against potential attackers.

In terms of data encryption systems, there exist some wide-spread solutions such as TrueCrypt[6] or eCryptFS[7] that usually ensure only local protection and are therefore not suitable for distributed cloud storages. Further proposals such as the one introduced by Kamara and Lauter (Kamara and Lauter, 2010) rely on token based searchable encryption schemes which causes a dispensable huge overhead.

In contrast, the *Sec*$^2$ middleware concept allows an efficient, seamless and fine grained data encryption on client devices. Therefore, trust relationships between mobile clients and cloud storage services are not necessary since content is protected before leaving the device.

Securing the wire can be done by using TLS (Dierks and Rescorla, 2008) or IPSec (Kent and Seo, 2005) which will perfectly fit for this purpose. However, encrypting the transport medium is just half-way towards secure document storage. Since storage means making data persistent it is insufficient to only focus on the way of deliverance. Durable encryption concepts for data storage have to be taken in account. At least when it comes to flexibility and hybrid documents, with encrypted and plaintext parts combined together in one single document, the introduced technologies such as Truecrypt reach their application limitations. These technologies are designed for different usage scenarios and thus are not appropriate for partial encryption.

## 3 SECURITY REQUIREMENTS IN DISTRIBUTED CLOUD STORAGES

Without a satisfactory level of security, users lack motivation to subscribe to cloud storage services. Thus, security is one of the main boundaries which still prevents a service from wider deployment. On the other hand, a security system must also deal with efficiency and real-time capability requirements.

We have identified the following goals that a security system must cope with:

**User Transparency.** The confidentiality of the data at the cloud storage and during the transport should be kept as transparent as possible to the user.

**Seamless and Cost-effective Integration and Installation.** The costs for integrating the security system at both, client and network side, should be as low as possible. Besides, the system should seamlessly operate and not interfere with parallel running applications

**Efficiency.** The security system should be based on

[2]http://sharepoint.microsoft.com
[3]http://www.opentext.com
[4]http://www.brainloop.com
[5]http://oceanstore.cs.berkeley.edu
[6]http://www.truecrypt.org
[7]https://launchpad.net/ecryptfs

```
<Document>
 <EncryptedKey Id="EncKeyId">
  <EncryptionMethod Algorithm="...xmlenc#rsa-1_5"/>
  <CipherData>...</CipherData>
 </EncryptedKey>
 <EncryptedData Id="enc">
  <EncryptionMethod Algorithm="...xmlenc#aes128-cbc"/>
  <CipherData>...</CipherData>
 </EncryptedData>
</Document>
```

Figure 1: Example of an XML Encryption metadata block.

light-weight schemes. Required resources should be kept minimal.

**Message Security.** As a security goal for messages our approach focuses on confidentiality.

**Secure Key Storage.** It must be possible for trusted code to securely store persistent data so that their confidentiality and integrity can be assured.

**Scalability.** The overhead of any security scheme targeting the emerging cloud storage technology should satisfactorily scale by an increasing number of users and by a growing amount of stored data.

**Tagging of Encrypted Data.** Although the payload of documents stored in the cloud is content protected by XML Encryption, the documents have to remain searchable. It should remain possible to some users to look up for a particular information without knowledge of a particular key. Searching particular information can be realized by including meta data on top of the encrypted payload.

## 4 FOUNDATIONS

### 4.1 XML Encryption

Since XML documents often contain confidential and reliable data, the W3C consortium developed standards that describe the XML syntax for applying cryptographic primitives to arbitrary XML data: XML Encryption (Eastlake et al., 2002) and XML Signature (Eastlake et al., 2008). In comparison to other security standards such as PGP (Elkins et al., 2001) or S/MIME (Ramsdell, 2004), XML Encryption and XML Signature can be applied to arbitrary data in the document. Therefore, these security standards offer high scalability and fine granularity of secured data.

Figure 1 gives an example of an XML message containing a ciphertext. This message consists of two main parts: The `<EncryptedKey>` part includes the symmetric session key. The `<EncryptedData>` part contains the payload data encrypted with the symmetric key.

### 4.2 SAML

The Security Assertion Markup Language (Cantor et al., 2005) represents a framework for modeling and defining security assertions (i.e., SAML assertion), authentication, and authorization data. The framework is XML-based and thus highly extensible. Included in the specification are means to use predefined protocols covering typical communication sequences. Moreover, the users can define their own assertions according to their needs and model new SAML-powered protocols. In general, assertions define statements about particular subjects. These statements may address the subject's identity, authorization, or additional attributes and may be interpreted by other parties.

## 5 *SeC*$^2$: THE PROPOSED SOLUTION

This section describes the *SeC*$^2$ system architecture and its operational details, focusing on the illustration of the security architecture.

Please note that in our concept we neither analyze the trust relationships between the clients nor address the infrastructure necessary to manage permissions. We assume that the clients are registered on the Key Server by a key administrator and are in possession of valid keys, which give them rights to access the desired documents. Moreover, during our description we assume the mobile device as trusted and free of malware, which could steal the decrypted data.

### 5.1 System Architecture

In order to fulfill the application developers' and users' requirements described in previous sections, we designed a modularized *Sec*$^2$ system architecture. This architecture is depicted in Figure 2 and consists of multiple independent components decoupled from each other. This high modularization enables to benefit from the most possible flexibility and changeability of components. The contents of the *Sec*$^2$ system are discussed in the following.

**XML Encryption Engine.** This unit is responsible for en- and decryption of XML payload and key data by relying on the concept of XML Encryption (Eastlake et al., 2002). To fulfill its purpose, the component utilizes components such as a crypto library and a microSD Card.

**Key Manager.** To keep track of keys already gathered and their usage in the en-/decryption process, the Key

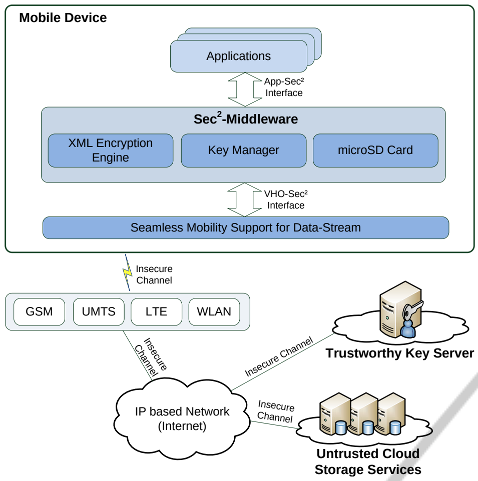



Figure 2: *Sec*$^2$ application scenario is based on three parties: Trustworthy Key Server, Untrusted Cloud Storage Services, and a mobile device. The encryption processing and key management within the mobile device is handled by the so called *Sec*$^2$ Middleware.

Manager stores a list of available keys. These keys are stored unextractable on the microSD Card.

**microSD Card.** It provides a highly secure storage for confidential key data such as the asymmetric private and symmetric secret keys as well as accelerates cryptographic operations. In our solution the microSD Card functions as a kind of trust anchor securely encapsulating key material unaccessible for unauthorized persons. Stored key material is not allowed to leave the device. In technical terms the used microSD Card represents a conventional SmartCard with additional storage capabilities. In our project we use the *Mobile Security Card SE (Standard Edition) 1.0* manufactured by *Gieseke & Devrient*[8].

**Trustworthy Key Server.** A major component, and the only external one that a client has to trust, is the Key Server. This server stores *all* cluster keys (discussed in 5.2), extractable only in encrypted form, and is able to provide them to authorized clients. Each cluster participant may request for an associated key (see 5.2, for details), which will only be delivered encrypted.

**Seamless Mobility Support.** To profit from seamless communication and to guarantee secure communication, we integrate a *Client Based Secure Handoff Solution* (CSH-MU) proposed by Tran et al. (Tran et al., 2010) into this module. Thereby, our framework supports a seamless service provisioning across a heterogeneous network consisting of UMTS/HSPA and WLAN. If one of the networks fails during the transmission process, another network link seamlessly continues with the transmission.

**Untrusted Cloud Storage Services.** As place for data storage, existing cloud storage providers can be used without adjustment or further extensions. The lack of trust relationship between service provider and client presents no issue since all the confidential data stored on this service is strictly encrypted and thus unusable for potential attackers without key knowledge. Moreover, the fine granularity of the XML Encryption allows to encrypt arbitrary document data. Therefore, the client can decide which data is confidential and must be encrypted. The plain text data can afterwards be used as metadata for queries.

As can be seen the security handling components (XML Encryption Engine, Key Manager, and microSD Card) are integrated in a central component called *Sec*$^2$ middleware. It should be noted that the security system can be separated from other modules and reused in many applications. The developers implementing their applications do not have to care about the key management or data encryption and decryption; they only have to integrate the *Sec*$^2$ middleware between their application and the mobility layer. Thus, the users' data can be seamlessly encrypted, securely sent over the insecure channel, and stored on the untrusted cloud storages.

## 5.2 Multi Stage Key Concept

In order to cover all the proposed usage scenarios in a secure manner, different key types are introduced. Each key type is used for a special purpose to provide confidentiality beyond user boundaries. The concept empowers the usage of commonly shared keys between legitimate participants. In particular the approach suggests three different keys as illustrated in Figure 3r:

**(asymmetric) Private/Public Keypair.** Each participant of the system owns a private/public keypair. These keys are used for proper authentication and secure key transport. The public key is deposited at the Key Server and the private key is stored in the secure part of the microSD Card at client side. Since public keys are not confidential, the Key Server may store these type of keys in plaintext in a database or lookup table.

**(symmetric) Cluster Key (CK).** A cluster key is a symmetric secret key that protects the particular cluster domain it belongs to. Users authorized to use the cluster key (and thereby authorized to decrypt all the documents in the cluster) are stored in an authorization table on the Key Server. Compromise of this key leads to a major break of confidentiality for all files

[8] www.gd-sfs.com/the-mobile-security-card/mobile-security-card-se-1-0/

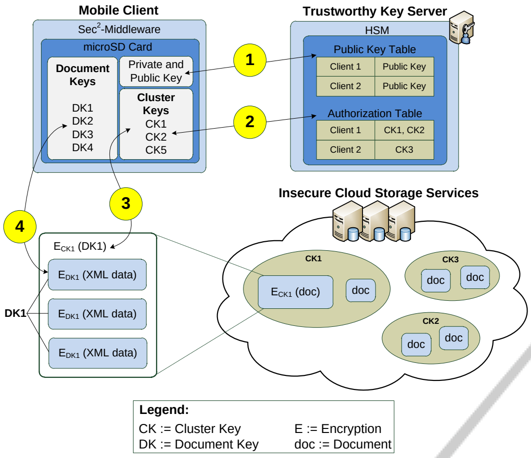


Figure 3: Our key concept is based on three key types: private/public keypairs belonging to the clients, cluster keys (CK) used for encryption of documents within specific clusters, and document keys (DK) for encryption of the documents.

protected by the associated cluster domain.

**(symmetric) Document Key (DK).** In order to protect the documents stored in the cluster, the cluster keys encrypt document keys. The document keys encrypt the content of the XML documents and thereby the payload of the cluster domain. Hereby, every document has its own document key, so that in case of compromise of a single document key, only the particular XML file containing this key is concerned. All other documents of the cluster domain remain safe. To achieve the highest security level, the document keys are also kept inside the secure key storage and are not extractable.

The usage of different keys provides a major advantage in contrast to the native approach of deliverance of particular keys for documents: Each participant needs effectively only a cluster key belonging to one's cluster domain to decrypt all the files protected by this domain (cluster). Since all documents contain their individual encryption keys, a cluster domain participant is able to decrypt the document parts by decrypting the document key and use it for decrypting the desired document parts.

Splitting key accountabilities aims to minimize the impact of key compromises. Adversaries who manage to take possession of a document key are not able to break the whole protection domain.

The described key concept offers a scalable key and user authorization management. It allows simple creation of new clusters and assignment of rights for new users. Additionally, the high scalability of the XML Encryption standard makes it possible to encrypt the document keys with different cluster keys. Therefore, XML documents belonging to different clusters can also be implicitly created. Combining all keys results in a key hierarchy that includes an integrated authorization concept. Only authorized users – users belonging to the cluster domain – can obtain the CK which is necessary to gain access to DKs. DKs are used to provide confidentiality of the several document sections. Authorization at the key server is done by XML signature generated with the user specific private key.

Our security concept relies on a few corner stones: **(1)** Security of the Key Server and local Key Storage, **(2)** Security of the used encryption and signature algorithms, **(3)** Trust to all cluster members, **(4)** Proper user-to-group tables at the Key Server.

### 5.3 Security and Usability Goals

The *Sec*$^2$ concept addresses a couple of major security goals which will be briefly discussed in the following:

**Confidentiality.** This goal is being achieved by using XML Encryption secured documents. The content is only accessible by those who are aware of a valid key.

**Authenticity.** To uniquely identify and authenticate users when trying to obtain cluster keys it is essential to provide means of secure authentication. In particular authentication at the Key Server side is done by providing signed SAML requests.

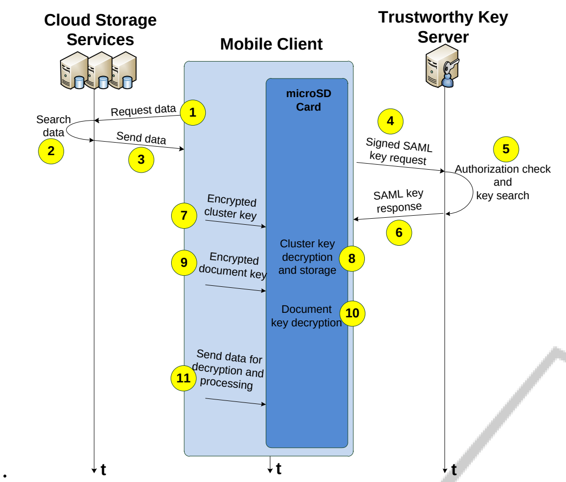
**Reliability (at client side).** Best possible service continuity, and thus reliability at client side, is realized by the Vertical Handoff Module which manages to keep the connection seamlessly alive beyound media boundaries.

**Integrity (optional).** An optional integrity protection can be achieved by applying XML Signatures. Secure and best practice applications are e.g. discussed by (Jensen and Meyer, 2011). The usage of Message Authentication Codes may also be a way to provide integrity.

**Tagging of Encrypted Data.** In order to provide the users with efficient searching abilities, only parts of the stored data are encrypted. The plaintext parts are used for searching purposes. Please note that we do not use the term searchable encryption as it rather implicates a cryptographic approach.

## 6 A PROOF OF CONCEPT SCENARIO

In order to make the idea of our concept clear we introduce a concrete communication process involving all the introduced parties from the previous section.




.
Figure 4: Communication process sequence of $Sec^2$ key concept.

Afterwards, we show the data storage and their structure using a simple calendar application with note functionality.

## 6.1 Communication Process Example

This section describes a simple communication procedure example between the Cloud Storage Service, Key Server, and the client attempting to get XML data. We assume that the client is in possession of a key pair, which is registered at the Key Server, and is authorized to access the cluster (data within the user group).

The communication example is depicted in Figure 4 and consists of the following steps (for the simplicity we depict only the microSD Card module inside of the $Sec^2$ middleware):

1. First, the client sends his search request with public metadata to the Cloud Storage Service.
2. The Cloud Storage Service evaluates the request and searches for the XML data.
3. The located XML data is sent to the client.
4. After the data arrives, the $Sec^2$ middleware extracts the cluster key information and creates a SAML request for the appropriate cluster key. Then, it signs the SAML request using the XML Signature standard with its private key (stored on the microSD Card) and sends it to the Key Server.
5. The Key Server verifies the signed SAML request and authorizes the client. Afterwards, it encrypts the requested cluster key and inserts it to a newly created SAML response.
6. The Key Server sends the generated SAML response to the client.
7. The $Sec^2$ middleware extracts the encrypted cluster key and provides it to the microSD Card.
8. The microSD Card decrypts the cluster key using the client's private key and stores it for subsequent processing.
9. The $Sec^2$ middleware extracts the encrypted document key from the already gathered document and forwards it to the microSD Card.
10. The microSD Card decrypts the encrypted document key using the cluster key.
11. As the microSD Card is in possession of a valid document key, the $Sec^2$ middleware can now use it for data decryption. Afterwards, the decrypted data is sent to the client application for further processing.

In order to achieve the highest security level, the cluster key cannot leave the microSD Card and all operations using this key must take place within the module. Once the client's microSD Card is in possession of a valid cluster key, the encryption process is straight-forward and the steps 4–8 can be skipped.

## 6.2 Calendar Application with Note Functionality

A calendar application with note functionality serves as a proof of concept scenario that will be briefly introduced. The demonstration platform on mobile devices side is Android[9] and the payload is stored at a cloud storage service providing XML database.

By using this proof of concept application, one is able to manage events such as meetings with notes in a secure manner. The confidential data of the events written in a note (in this case the event description) is encrypted, whereas date, time, and additional location are left unencrypted for searchability and collision detection purposes. After entering a PIN, the user can write notes for a certain event. Confidential parts of the note are marked for encryption and are, therefore, only visible for authorized cluster groups.

Given this scenario, users outside the cluster can detect event and location collisions, but confidential event details are only visible to cluster members.

A data flow example for decryption between the Application, $Sec^2$ middleware, and cloud storage is illustrated in Figure 5. In this scenario, the queried event data is delivered to the client. The $Sec^2$ middleware decrypts the confidential data, allowing the client application to process them in a decrypted representation. After the client's processing is done, the

[9] http://www.android.com

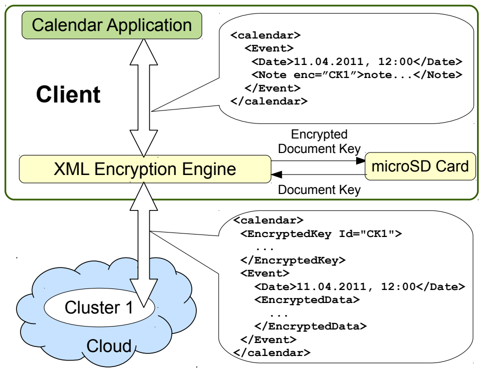



Figure 5: An example data flow.

data is sent again through the middleware to the XML server.

In order to encrypt the appropriate data before sending to the XML server, the data dedicated for the encryption process has to be indicated. For this purpose, the application adds specific attributes to the elements that will be encrypted. In our example, we use the attribute `enc="CK1"` meaning that the marked part of the note must be encrypted with cluster key *CK1*.

## 7 CONCLUSIONS & FUTURE WORK

The proposed solution provides a convenient way towards secure data storage in the cloud without the necessity of trust relationships between the storage provider and the mobile client. Furthermore, the *Sec*$^2$ concept integrates transparent into existing architectures with little to no changes concerning cloud infrastructures. Control over stored data is given back to the cluster domains, instead of relying on trust relationships and cloud storage encryption.

With upcoming steps, we are researching on options for extending the concept to a scenario without the need of a trustworthy Key Server.

## REFERENCES


Cantor, Kemp, Philpott, and Maler (2005). Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Technical report.

Dierks, T. and Rescorla, E. (2008). RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2. Technical report, Network Working Group.

Eastlake, Reagle, Imamura, Dillaway, and Simon (2002). XML Encryption Syntax and Processing. Technical report.

Eastlake, Reagle, Solo, Hirsch, Roessler, Bartel, Boyer, Fox, LaMacchia, and Simon (2008). XML Signature Syntax and Processing (Second Edition). Technical report.

Elkins, M., Torto, D. D., Levien, R., and Roessler, T. (2001). MIME Security with OpenPGP. RFC 3156 (Proposed Standard).

Jensen and Meyer (2011). Expressiveness Considerations of XML Signatures. In *SAPSE 2011: IEEE International Workshop on Security Aspects of Process and Services Engineering*.

Kamara, S. and Lauter, K. (2010). Cryptographic Cloud Storage. In *Financial Cryptography and Data Security*. Springer Berlin / Heidelberg, Berlin, Heidelberg.

Kent, S. and Seo, K. (2005). RFC 4301 - Security Architecture for the Internet Protocol. Technical report, Network Working Group.

Molnar, D. and Schechter, S. (2010). Self hosting vs. cloud hosting: Accounting for the security impact of hosting in the cloud. In *Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS)*.

Ramsdell, B. (2004). Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification. RFC 3851 (Proposed Standard). Obsoleted by RFC 5751.

Tran, T., Yousaf, F. Z., and Wietfeld, C. (2010). Cshmu: Client based secure handoff solution for mobile units. In *The 21st IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC)*, Istanbul, Turkey. IEEE.

Wang, S.-C., Yan, K.-Q., Liao, W.-P., and Wang, S.-S. (2010). Towards a load balancing in a three-level cloud computing network. In *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, volume 1, pages 108 –113.

Wu, J., Ping, L., Ge, X., Wang, Y., and Fu, J. (2010). Cloud storage as the infrastructure of cloud computing. In *Proceedings of the 2010 International Conference on Intelligent Computing and Cognitive Informatics*, ICICCI '10, pages 380–383, Washington, DC, USA. IEEE Computer Society.

Zhang, S., Zhang, S., Chen, X., and Huo, X. (2010). Cloud computing research and development trend. In *Proceedings of the 2010 Second International Conference on Future Networks*, ICFN '10, pages 93–97, Washington, DC, USA. IEEE Computer Society.


## APPENDIX

Our work has been conducted within the *Sec*$^2$ project[10], which is funded by the German Federal Ministry of Education and Research (BMBF).

We would like to thank Benedikt Driessen and Florian Kohlar for their contributions.

[10]http://www.sec2.org