# Secure Data Storage and Exchange with a Private Wallet

Oliver Jäger[1], Frank Kramer[2], and Bernhard Thalheim[2]

[1] NTT DATA Deutschland GmbH, Hammerbrookstrasse 89, 20097 Hamburg, Germany
[2] Christian-Albrechts-University Kiel, Computer Science Institute, 24098 Kiel, Germany

**Abstract.** Sharing private data between various users is a daily scene. The owner has both rights at the same time, the right of a self-determined and the right of a policed usage of his private data by foreign holders. We develop an approach that allows an owner to store and share his private data and gain full control about the usage of his private data. Therefore, we develop and implement a flexible system that uses a peer-to-peer architecture and modern encryption standards to realize data privacy. We call this system *private wallet*.

**Keywords:** privacy, communication, security, data storage, data management.

## 1 Introduction

The self-determination of privacy is a highly-valued asset and must be protected. The Universal Declaration of Human Rights of the United Nations says in article 12: *No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks* [1]. Also, other national laws see the self-determination of privacy as a high priority. For example, the Basic Law for the Federal Republic of Germany, the German constitution, defines in article 2 *that every person has the right to free development of his personality insofar as he does not violate the rights of others or offend against the constitutional order or the moral law* [2]. Today, the web makes it possible, to get information about a person very fast. Hence, the self-determination of personal data must be taken under the consideration of privacy. For example, the scientific article [3] shows what happens, if we abandon our data privacy. It describes, how easy it is for somebody to get information about a person only by scanning an image of this person.

To evolve a system to secure private data, it is important to distinguish between the owner and the holder of data. A person is owner of data, if he has created this data. Therefore, an owner can read, change and delete his data at will. Furthermore, only with his permission other persons can use his data and the owner is informed about how his data is used and changed from other



N. Bassiliades et al. (eds.), *New Trends in Database and Information Systems II*,
Advances in Intelligent Systems and Computing 312, DOI: 10.1007/978-3-319-10518-5_5

persons. A person who uses the data of an owner is a holder of data. He can only use the data with the permission the owner has given him for the data. Moreover, the owner must inform the holder, if he has changed something on his data. Hence, a system is needed that allows an owner permanent control of his private data against a holder. The control of the private data covers, on the one hand, the encryption of the data, and on the other hand, the best possible logging of all usage of the data.

This paper presents an approach to reach such a data control. Therefore, we develop and implement a concept called *private wallet*. Every owner gets his own wallet to store his private data inside. To secure the data in the wallet, all data is only stored encrypted and solely the owner knows the key. Furthermore, the *private wallets* are connected within a network. The owner of the data can exchange his data with other users within the network. For every exchanged data the owner defines the permissions a holder can get. Additionally, all usage of the holder is logged for the owner. Consequently, an owner is informed permanently which holder uses his data in what way. In our approach, section 2 will first present the functionality of our *private wallet*. We will present our six main processes that are realized in our approach. Section 3 will then describe, how we realize our privacy wallet. This covers solutions for ground functionality, user management, encryption and data storage. In section 4, related work will be presented. We will take a look on other solutions that exist to secure private data. Finally, a conclusion and a short outlook on future research will be given in section 5.

## 2 The Privacy Wallet Functionality

This section presents the functionality of our *private wallet* approach. There are six different business use cases for the *private wallet*. Each business use case can be regarded as an independent process. All transfers between pares within a process are asynchronous transfers. We will describe each business use case. All these use cases can be modelled with a business process modelling language, for example BPMN 2.0[4]. Due to the limitations of this paper, we present only one process model as an example. We refer to the Bachelor thesis of [5] that presents the other process models too.

### 2.1 Request-a-Key

The first functionality we want to present is *Request-a-Key*. With *Request-a-Key*, the key for decrypting a specific document is requested by a user to the system of the key owner. Figure 1 presents this use case as BPMN model.

In the owner's system, a check is carried out in the keystore for whether the specific document and the specific user, who request the key, exist. If no key exists in the keystore, the fact that there was an incorrect request from this user for this document is logged. Reasons could be that the user has been denied rights to the document or the user received this document without authorization.

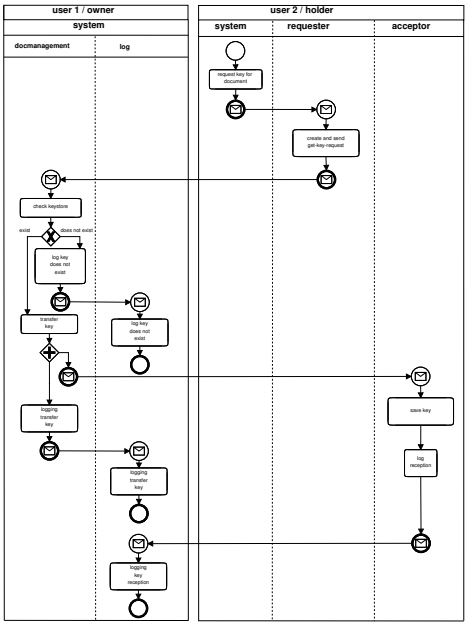


**Fig. 1.** BPMN request-a-key

If a key exists in the keystore, this key is transferred and the transfer of the key is logged in the system of the owner. The user's wallet receives the key and stores the key to the document. The reception of the key is confirmed. The wallet of the owner stores the receipt confirmation in a log.

### 2.2 Request-a-Document

With *Request-a-Document*, a particular document in the owner's system is requested. First, a connection is established to the desired participant. This connects both *private wallets*. The requester identifies itself on the basis of authorization features. The *private wallet* receiving the request now shows the requesting *private wallet* all documents to which the requesting *private wallet* is permitted to have access. It does not matter whether the requesting *private wallet* has full access or read-only access to the document. The requester can now select from this document list the documents that are to be transferred into his own *private wallet*. These documents are then sent by the owner to the requester. Each receipt must be confirmed. This receipt confirmation is then stored by the wallet which sent the documents.

### 2.3 Response-a-Document

With *Response-a-Document*, a particular document is sent to a particular person from the owner's system. To do this, the owner chooses the document to be sent in his own wallet. The document is encrypted for the selected person and sent to him. The key is also stored in the local keystore of the owner. The document sent is automatically written to the wallet of the recipient. Once this has happened, the document is displayed in the document list and a receipt confirmation is sent to the sender. The sender, that is, the owner of the document, receives this receipt confirmation and the system stores it automatically.

### 2.4 Synchronize

With *Synchronize*, updated data are sent to other users. These data cover new files, people, person objects, or altered document objects. Therefore, the wallet of the user that has changed the data starts the synchronization. The other wallets receive these changes and update the data in the wallet in the background on a fully automatic basis. Thereby all wallets only synchronize from the wallet that has started the synchronization. The user himself does not need to take any action. If a wallet cannot be reached, synchronization is repeated later. Thus, the synchronizing wallet looks in a periodical time, if the missed wallet is online, and then sends the changes. This ensures that each participant of the group receives all updates needed.

### 2.5 Ask-for-Extended-Rights

With *Ask-for-Extended-Rights*, a holder of a document requests from the owner an extension of the rights to this document. In our case, extended rights mean an extension of existing rights. For example, if a holder has only read rights on a document, he can ask for edit rights for this document. The owner must actively decide whether to grant the requested rights or not. If the owner does not grant the requested rights, the request is logged with the negative decision. If the owner grants the requested rights, the document is re-encrypted and transferred to the holder. In addition, the file transfer is logged and the new key is stored in the keystore. The wallet of the holder automatically receives the file and stores it in the filestore. In the process, the old file is overwritten and the old key, if it exists, is erased. After successful receipt of the document, a receipt confirmation is sent to the owner. The wallet of the owner saves the receipt confirmation in the form of a log.

### 2.6 Request-a-Person

If a new user joins a group, it is necessary that all members of the group know this new user. In the peer-to-peer network, the new user is known in the form of his email address if he is online, but other information is missing, in particular the public key. If a new user now participates in the group, his complete data

is automatically distributed by the system to all participants who are online. To reduce administrative effort, that is, the management of the information regarding which users have been informed and which have not, the principle of an obligation to provide information is replaced by an obligation to retrieve information in this case. If a *private wallet* sees that an 'unknown' user is in the group, that is, in the group online, then this *private wallet* automatically requests the data of the 'unknown' user from the *private wallet* in the background. This data is then sent back to the requesting *private wallet* on a fully automatic basis.

In contrast to the other stories, a check for whether the data actually arrived is not performed with this process. This is not necessary, because no security and control mechanisms are disrupted or undermined if the data is not present. Sending of the documents can only take place if the personal data is locally available. If problems occur with the transmission of personal data, the wallet simply tries again. To make sure that there was really a receipt confirmation in the operations *Request-a-Document, Response-a-Document, Request-a-Key* and *Ask-for-Extended-Rights*, a check is carried out. After an appropriate time, the system checks whether there has been a receipt confirmation. If this is the case, then everything is fine. If this is not the case, this document is blocked for the receiver. This happens when a delete command is sent to the receiver wallet and at the same time the entry is removed from the keystore. This process is logged.

After presenting the functionality, the next section will describe the concrete realization of the *private wallet*.

## 3 Privacy Wallet Realization

This section describes the realization of the *private wallet*. The *private wallet* serves as an intelligent interface. This interface can be placed over a document management system (DMS), for example, but it can also work on a standalone basis. This makes it possible to use it in a very flexible manner. A set of wallets communicate in a peer-to-peer network. The whole communication is secured with a hybrid encryption approach and an additional usage of steganographic functions. Figure 2 shows this general structure of such a *private wallet* peer-to-peer network.

The next sections will take a closer look into the construction of such a *private wallet*. To see the implementation, we refer again to the Bachelor thesis of [5].

### 3.1 Modules and Task Areas

The implementation of the *private wallet* is modular. This allows easy replacement of individual components, such as another encryption or another communication framework. The following basic functional components are implemented for the *private wallet*.

**Importing Documents.** The *private wallet* is a closed system. Only within this closed system can the built-in safety mechanisms be effective. Therefore,

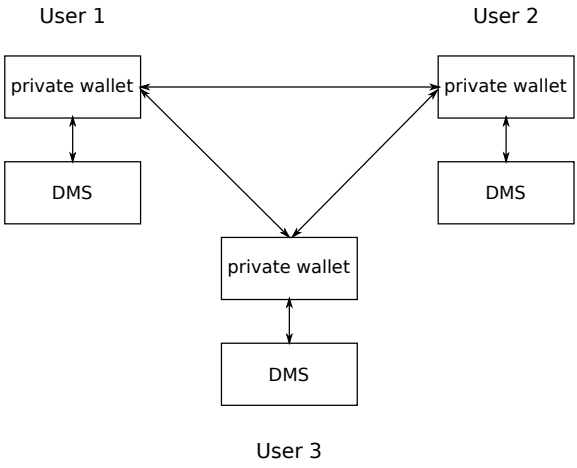


**Fig. 2.** Peer-To-Peer *private wallet* Model

a function must be provided that allows you to import data into this system. For this purpose, the *private wallet* makes the *Import* function available. Data such as document ID, creation date, and author are generated by the system and automatically added. Other data must be entered by the user. The file is encrypted immediately and stored in the filestore. The associated metadata is stored in the database. From this point on, the document is available in the *private wallet*.

**Communication.** A major component of the *private wallet* is exchange among the participants. In this approach, the decision was made in favour of a peer-to-peer network. In [6] a peer-to-peer system is described as a self-organizing system that is composed of autonomous units called peers. All peers have equal rights in the system and all peers use their resources and services completely decentralized. This short description can be extended by a set of characteristics of a peer-to-peer network as described in [7]. An ideal system has all these characteristics. But in most cases only a subset of these characteristics can be found in a peer-to-peer system. Using a peer-to-peer system brings some advantages over a client-server system. Through the omission of a server, the vulnerability of the entire system is reduced. Another advantage of a peer-to-peer network is simple distribution of information. It offers the possibility of sending a message to individual members of the group or to all members at the same time. Communication is always encrypted. This ensures that even if an outsider infiltrates the group, the information is secure.

**Logging.** Any action on or with a document triggers notification of the owner and author. To ensure this, the information about such an action is kept in a log object and sent to the *private wallet* of the owner and author. The owner and

author stores this log object in his overall log. If one of the two participating *private wallets* is offline or unreachable, this log object is stored in the sender wallet and marked as not sent. As long as the sender wallet is turned on, the system attempts again and again to transfer non-transferred log objects at appropriate intervals until a transfer is successful. A log entry contains the document ID in your own *private wallet*, timestamps of when an action took place and when this log entry reached your own *private wallet*, the type of use of the document, the status, how many times this action was tried, and the user and his IP address. But our logging is limited. If a holder copies a document or takes a screenshot, we have no logging about what he is doing with the document any more. So at this point, we lose the control over an owners document.

**Distributing.** The aim of the *private wallet* is to exchange data securely and to keep as much control as possible over the exchanged data. This makes it necessary to distribute data. This data are documents or document data, keys, and log entries.

### 3.2 User Management

The *private wallet* requires its own user management, because only if users are registered in the *private wallet* network, they can exchange data. Therefore, in this user management, it is defined for each user who he is, in what way to communicate with him, what type of encryption is used, and to which user group he belongs. Allocation to user groups offers the possibility of easily assigning keys and rights to documents to several users. In this way you have the option to automate certain approvals for the use of documents.

### 3.3 Encryption

As described in [8] there are two different approaches for secure communication, steganography or cryptography. In our implementation a hybrid of two types of encryption is selected for cryptography. Symmetrical encryption is chosen for encryption of the document. In this way, it is possible to benefit from the speed advantage over asymmetrical encryption. The downside in terms of the effort and lower security with the key exchange is balanced out by asymmetrical encryption of the key of the symmetrical document encryption. A 128-bit key with the AES cryptosystem [9] is used for document encryption. A 192-bit or 256-bit key is also possible. For the RSA encryption [10], 2048 bit is used. Adequate security is already provided with a smaller number of bits. Because RSA encryption is only used to encrypt the AES key, meaning the text to be encrypted is very small, there are no performance drawbacks, but much more security.

Furthermore, this *private wallet* offers the possibility of increasing the security of the encryption through steganography. The encrypted key is hidden in an image. Hence, an owner can hide the key by using a special function from the private wallet. A holder can only reveal the key from the image by using the same

function in his private wallet. Due to the combination of RSA encryption and steganography, the security is classified as very high. The method used in this *private wallet* for steganography is very simple. The method can be exchanged by every other known steganography algorithm. Here, every image is composed of colour points or pixels. A pixel has a red value, a green value, and a blue value. These values are between 0 and 255. To hide a message in the image, in each pixel each colour channel is changed in the last bit. The decision whether the last bit is set to 1 or 0 depends on the message to be hidden. For this, the letters of the message are converted into the associated ASCII values. These ASCII values are converted into binary form. Take as an example that DB is hidden. The ASCII value for D is 68, and the value for B is 66. In binary terms, it is the values 01000100 for decimal 68 and 01000010 for decimal 66. With the binary values, it is necessary to ensure that the leading zeros (always eight digits) are taken into account. Taking into account that it is always only the last bit of a colour channel that is changed and a pixel has three colour channels, eight pixels are needed to hide two letters and the message end character (ASCII 0). The number of pixels required is given by the following formula:

$$\lceil numberofpixels \rceil = (numberofcharacters + 1) * 8/3$$

The bit sequence of the letter combination DB results in 0100010001000010, which is 16 bit values. Now the the bit values are entered in sequence into the most recent bit of a colour channel in each case. If the last bit of a colour channel is identical to the bit entered, no change takes place. In this example, the last bit of the first colour channel of the first pixel receives the value 0, the last bit of the second colour channel receives the value 1, and the last bit of the third colour channel receives the value 0. Since all three colour channels have now been used, the next pixel is used. This is repeated until all bits of the bit sequence and the message end character have been entered. Since only the last bit of a colour channel is manipulated, the overall colour of a pixel is changed by 1/256. This is only 0.39 percent. As a result, the colour change for each pixel is from 0 to 0.39 percent. On average the change is about 0.2 percent. This change is not discernible to the human eye. This means the message has become invisible.

### 3.4 Data Transfer

There are various possibilities for transferring the data between the *private wallets* within the network. Using TCP and UDP, data can be transferred from both computers and mobile devices such as smart phones or tablets. This works both in a local network and over the Internet. The advantage of this transfer option is that all systems have this possibility of use.

Transfer via Bluetooth or NFC is suitable only for mobile devices such as smart phones or tablets. However, this type of transfer offers great advantages in terms of security. Since NFC[1] or Bluetooth[2] only work over short to very short

[1] range: up to 10 cm.

[2] range: class 3 approx. 10 m outdoors, class 2 approx. 20 m outdoors, class 1 approx. 100 m outdoors.

distances, the probability that such a connection will be 'overheard' is close to 0, since it is immediately apparent if someone is hanging around in the vicinity.

Transfers over a local network or the Internet are suitable due to the high transfer rates both for the exchange of documents and for the exchange of keys. Networks have transfer rates up to 1 gigabit/s, and the Internet provides transfer rates of between 1 and 100 Mbit/s per second for private households.

Bluetooth, on the other hand, only has transfer rates of 57.6 kbit/s to 732.2 kbit/s. Transfer of documents is possible here, but the advantages of Bluetooth come with smaller data packets, such as a key, because the security is significantly higher due to the proximity to the exchange partner. The same applies to NFC. With NFC the data transfer rate is only up to 424 kbit/s.

### 3.5 Data Management

To store the data, every *private wallet* gets its own small database. Every database get the same database schema to store the data. Figure 3 shows this database schema as Higher-Order Entity-Relationship Model (HERM)[11].

The central element is the document. A document can have one owner and multiple holders. The owner is the person who owns the document. This is in most cases the author of the document. The holder is the person who has power or control over the document. A document can have each number of keys with the corresponding rights. Behind each key is a holder with the appropriate rights. For each document, there is metadata. Information is stored for every person who participates in the *private wallet*. The log contains any number of log entries. A log entry provides information about the affected document, the point in time of the activity, the actual process, the status (the action was successful or not successful), the number of attempts, and the IP address of the device on which this action was performed. In the *private wallet*, users can define the document types in any way required. This ensures a high degree of flexibility.

Throughout the application, work is carried out only on data objects that are already stored in the database. What made the decision for such an approach was the objective of not losing any data. This is particularly important for guaranteed logging of actions on documents. For some operations, it is necessary to update existing data in the database. Only the following personal data can be changed: first name, last name, the marker, whether it is a local user or a group member, the password for local login to the *private wallet*, the public and private key, the document data rights to a document, the document key, and the expiration date. All other data, for example the email or the salutation, may not be altered under any circumstances.

Data which is stored in the database is deleted only by setting a marker which indicates that it is a deleted data record. The only exception is temporary data. Temporary data in this *private wallet* consists of log entries that have not yet been transmitted to the author of the document. They are removed from the database. Deleting data records only logically offers the advantage that no data inconsistency can occur. For documents marked as deleted, the physical file is deleted in the file system.

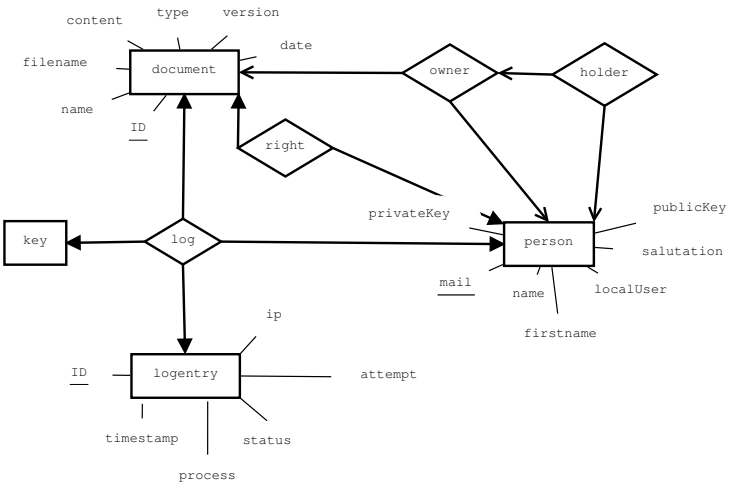


**Fig. 3.** HERM-Schema of the *private wallet*

## 4 Related Work

There exists a great set of implementations in the area of secure data storage and exchange. Here, we want to describe some of these implementations that track concepts similar to our *private wallet* approach. First, we look at cloud services like Dropbox[12] or Apple iCloud [13]. There, a user can store data on a foreign server and can share these data with other users of the cloud. The main problem is that all data that is saved in these cloud services are stored on an third party foreign server. At this point an owner has no full control over the usage of his data by this third party. He must trust in the service that there is no illegal access to the data by the third party. Furthermore, an owner is dependent on the security of the cloud service system. In our approach, only holders who have the owner's permission can get access to data. If an owner shares data, this data is transferred from the owner system to the holder system. Thus, no third party server system for storage and transfer is used. Additionally, in most of the cloud services our data is not stored encrypted within the cloud. We must install special programs for encryption before storing the data into the cloud. In our private wallet approach every document is encrypted automatically when it is send to a holder. Only if the owner gives the holder the key permission, the holder can decrypt and use a document.

Next, we take a look at an application that is close to our approach. The *Deutsche Post AG* provides a software called *DocWallet* [14]. This software can be used for secure personal document management. Therefore, a user can import documents into the *DocWallet* system. Within the system, the document is end-to-end encrypted with AES 256 and synchronized with every system where the *DocWallet* software is installed. Hence, a user has access to his documents on all

his systems, including mobile devices like smart phones or tablets. All this is free of charge for the user. Additionally, one can pay for a premium account. Then the data within the *DocWallet* is synchronized with a cloud server in Germany that backups all imported files in the cloud. Unlike our *private wallet*, there is no data exchange between various *DocWallet* users. The system is only for single user data management.

Another mobile application for smart phones and tablets that can be used for a secure exchange of chat massages with attached videos or images between users is Snapchat [15]. Before a user can exchange a message with another one, he must define an expiration date for the message he wants to send. The receiver of the message can look at it on his device until the expiration date is reached. After this, the message is deleted from the receiver device. This should guarantee the securtiy of the videos and images that are send via Snapchat. But there are some problems with Snapchat. On the one hand, the receiver of a Snapchat message can take a screenshot of the received object. Then he has permanent access to the object without the control of the owner. On the other hand, user have to trust Snapchat that they fully delete the message. There are hints that this is not done by Snapchat [16]. Despite everything Snapchat is a first approach for a secure data exchange were the user has the control over how long another user has access to his data.

The last service we want to present is the data exchange service Mega [17]. Mega can be used to exchange data with other users through the internet. Therefore, a user can import any digital information to the Mega server. The information is end-to-end encrypted with AES 128 before it is stored on the server. Only the user gets the key to decode the data. Hence, no other user or Mega itself can decode the private data. If a user wants to share the information with another user, he has to exchange the key with this user, so that the other user can get access on the data too. How the user exchange the keys is not the task of Mega. Other than our *private wallet* the focus of the encryption of information is not the self-determination about the own data. The encryption of the data is a legal protection for Mega. Thus, they can assert that they never know what users send over their filesharing system.

## 5 Conclusion and Outlook

Exchanging data and protecting the privacy of data for an owner is not a contradiction. The approach outlined in this paper presents a system by which an owner of data can store and exchange his data with other users without loss of control. Therefore, we implement a system that is based on a peer-to-peer network to exchange data between users of this network. All exchanged data is encrypted with modern algorithms and only the owner of the data can decontrol the usage for different holders. Furthermore, the usage of steganographic algorithms and the storage of private data within a database increase the security of the private data once again.

We have only outlined a first approach of our *private wallet*. In a subsequent step, we have to finalize our prototype to a stable version for release. After this,

we can extend our system with additional features. One could be a user management that has not only user but also user groups in the system . Furthermore, we have only a version for personal computers at the moment. Hence, we have to implement a mobile version to ensure secure data exchange also on mobile devices such as smart phones and tablets. Finally, we have no recovery strategies for media break while using the *private wallet*.

## References

1. UN: The Universal Declaration of Human Rigths, http://www.un.org
2. Bundesrepublik, D., Dürig, G.: Grundgesetz. 44. Auflage edn. Dtv (2013)
3. Acquisti, A., Gross, R., Stutzman, F.: Faces of facebook: Privacy in the age of augmented reality. BlackHat, USA (2011)
4. OMG: Business Process Model and Notation (BPMN) Version 2.0, http://www.omg.org/spec/BPMN/2.0/PDF
5. Jäger, O.: Technologien für das Privacy Wallet (April 2013)
6. Oram, A.: Peer-to-Peer - Harnessing the power of disruptive technologies. OŔeillt (2001)
7. Steinmetz, R., Wehrle, K.: Peer-to-peer-networking & -computing. Informatik Spektrum 27(1), 51–54 (2004)
8. Singh, S.: Geheime Botschaften Die Kunst der Verschlüsselung von der Antike bis in die Zeiten des Internet. Carl Hanser Verlag München Wien (2000)
9. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
10. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21, 120–126 (1978)
11. Thalheim, B.: Entity-Relationship Modeling: Foundations of Database Technology. Springer (2000)
12. Dropbox, https://www.dropbox.com/
13. iCloude, https://www.icloud.com/
14. Deutsche Post: Docwallet, https://docwallet.de/
15. Snapchat, http://www.snapchat.com/
16. CHIP: Snapchat: Sexting-App löscht Videos nicht, http://www.chip.de/news/Snapchat-Sexting-App-loescht-Videos-nicht_63702600.html
17. Schmitz, K.: Mega, https://mega.co.nz/