TRANSACTIONS ON DATA PRIVACY 8 (2015) 89–111

On the Feasibility of (Practical) Commercial Anonymous Cloud Storage

Tobias Pulls*, **Daniel Slamanig****

*Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden.

**Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria.

E-mail: tobias.pulls@kau.se, daniel.slamanig@tugraz.at

Abstract. Current de facto standard payment and billing models for commercial cloud storage services provide a plethora of information to the cloud provider about its clients. This leads to a hostile environment when seen from a privacy perspective. Motivated by recently leaked facts about large scale governmental surveillance efforts as well as the lack of privacy-preserving measures in existing commercial cloud storage services, in this paper, we investigate the feasibility of so called *anonymous cloud storage* services which require user payment (which we call *commercial anonymous cloud storage*). Anonymity in this context can be seen as the absence of information to uniquely identify a provider's client that is storing and manipulating data at the provider while at the same time still allowing fair billing, for both, the clients and the cloud provider.

Although encrypting data prior to outsourcing helps to protect data privacy and can be achieved without the cloud provider's consent, the issues we are interested in, do not seem to be achievable that easily. However, while various measures for the latter issue, i.e., realizing access privacy, have been studied in the past, the role of privacy in context of billing and payment for cloud storage has, until now, remained unexplored. We therefore introduce an abstract model for commercial cloud storage services to define various types of anonymous cloud storage, study several payment and billing models for cloud storage services and their impact on the anonymity of the service's clients. Moreover, we discuss several solutions to provide anonymity within the different models.

Our findings highlight the importance of anonymous payment for the practical deployment of commercial privacy-friendly cloud storage services. Furthermore, we provide directions for future work in some settings, i.e., when anonymous payment is not available, as interesting open challenges.

1 Introduction

Central to the concept of cloud computing is the provision of infrastructure, platform, or software as a *service*. Using such cloud services often requires users to *pay*, either directly with money or indirectly by being subjected to commercial advertisements and profiling. Advertisements and profiling are used for instance by major free social networking services. In this paper, our focus is on *cloud storage services* that require monetary payment and the privacy issues, in particular with regards to anonymity, that emerge as a consequence of the payment requirement. Advertisement and profiling, often tightly coupled for efficiency reasons, have their own set of privacy issues [22, 8] but are out of the scope of our work in this paper.

When looking at current payment models for cloud storage services, the privacy of the services' users is usually not considered an issue. Users are required to register using personally identifiable information (PII) at a cloud provider to setup an account to which billing and payment information are connected. Consequently, the provider can monitor the storage consumption behaviour of its users, i.e., every store, read, and delete operation can be tracked, and the provider thus has access to a complete behavioural profile of its users. Recently, leaked facts about large scale governmental surveillance efforts[1], however, may require to provide users with the guarantee that such information are *not* available to a cloud storage provider.

Such aforementioned behavioural profiles are, however, usually required to create *bills* for the users, i.e., to determine *what* a user has to pay for. The payment model then determines *how* such a bill is settled, e.g., via a credit card. It is thus vital to consider the combination of billing and payment models at the provider. We note that the *same* issue exists with the vast majority of other cloud/online services, besides storage services. However, in other types of cloud/online services, anonymous subscription can typically be achieved more easily [9, 33] and adequate privacy can be provided.

Achieving adequate privacy protection within cloud storage services—in particular if they have fine-grained information about users' consumption behaviour—seems, unfortunately, to be extra problematic. By definition all data is stored at the cloud provider and store, read, and delete operations need to somehow be attributed to the respective user in order to allow billing and payment. Consequently, the stored data objects are an additional obstacle from the privacy point of view. The questions are whether and to what extent this can be realised while preserving the privacy of the users of such a service.

This paper initiates the study of anonymity within this setting, thereby seeking to rigorously investigate and answer these two questions. To the best of our knowledge, there is no similar work covering our concrete setting. We emphasize that this paper solely considers technical aspects of realizing anonymous cloud storage, but does not consider other perspectives such as user acceptance and policy, which clearly are not less important when it comes to practical systems.

1.1 Related Work

An important issue in cloud storage is the *data privacy* of outsourced data. Data privacy is actually one of the major concerns with cloud storage services, since the provider could accidentally or deliberately disclose the data, or use the data for unauthorised purposes. Protection against data privacy threats is typically achieved by means of data encryption [49], data sanitization [1], data anonymization [75] or applying various other concepts from statistical disclosure control [42]. We stress that data encryption actually is a sine qua non when outsourcing data to the (public) cloud, and in particular with respect to anonymity, as unencrypted data may trivially leak information about the data owner. Moreover, many recent concepts propose the use of multiple independent cloud storage services (aka distributed cloud storage or clouds of clouds) and distribute data fragments to the single provider (cf. [68] for an overview of various approaches).

Another line of work investigates measures to provide *access privacy* for outsourced data in the cloud, i.e., hiding access patterns for data objects. This is typically achieved by means of private information retrieval (PIR) [28, 35, 40], and when read and write operations should be private and indistinguishable by means of oblivious random access memory (ORAM)

[1] `http://www.theguardian.com/us-news/the-nsa-files`, accessed 2015-02-16.

[36, 37, 73, 71] or a combination thereof [41]. In context of access privacy, one may also be interested in realizing privacy friendly access control in the cloud, which can be achieved by various cryptographic means [13, 78, 59, 67, 47].

Untraceability and anonymity in context of Software as a Service (SaaS)—with a focus on web services—have relatively recently been investigated by *Pacheco et al.* [54]. In brief, their proposal consists of establishing a broker entity, a so-called Third Party Broker (TPB), which intermediates contracts between consumers and providers, and issues anonymous credentials that allow authenticated service usage and accounting. In another paper [55], *Pacheco et al.* also consider payment for SaaS consumption by using anonymous E-Cash, whereas the TPB has the additional role of issuing pre-paid credits in the form of E-Cash to users. However, we note that both aforementioned papers are very "high-level" and do not treat the anonymity provided by their approaches sufficiently.

In [66], *Slamanig* discusses anonymous yet accountable resource consumption and payment, e.g., for cloud storage services, and proposes an approach based on updateable anonymous credentials from Camenisch-Lysyanskaya signatures. Following this scheme, *Pirker et al.* [57] present a practical implementation of a framework for privacy preserving resource payment for cloud services on constrained devices such as smartphones.

In [43] *Ioannidis et al.* present a credential-based network file storage system with provisions for paying for file storage and getting paid when others access files, which may also be adopted to cloud scenarios. The approach is based on lightweight micropayments, but does not account for user privacy.

Billing in the context of cloud storage (and cloud computing in general) has also been investigated recently. Existing works thereby focus on verifiable resource accounting [63] meaning that cloud customers can be sure that they are billed what they have really consumed, as well as alternative and potentially more fair billing models, which take into account the real resources expended at the cloud provider [76].

In [30], *Danezis et al.* present an approach for privacy-preserving fine-grained billing within the differential privacy framework targeted to smart metering, but it may also be adopted for computation clouds (as suggested by the authors). The basic idea is that users add some, in the long run small, amount of noise (costs) to their bill. The true cost of service provision is tracked across billing periods, but not revealed to the service provider, which can only verify the deposited funds cover costs. However, although elegant, this model does not seem to be applicable to the cloud storage setting for various reasons. The main reason is, that the cloud provider receives all the requests and thus sees the exact consumption (bill) of the user and thus adding noise does not bring a benefit with respect to privacy.

1.2 Contributions and Organisation

The contributions of this paper are as follows. We define a model that covers important aspects of commercial cloud storage services and define what constitutes anonymous cloud storage in this model. Furthermore, we identify several billing models and variants for payment in the context of cloud storage services and investigate the relationship between anonymity, billing and payment. We highlight the importance of anonymous payment for wide acceptance and use of cloud storage services. This is a necessary but not sufficient prerequisite for providing anonymity in this context. We also identify *oblivious aggregated billing* and *verifiable shuffling of data* as interesting open challenges for future work for anonymous cloud storage.

The rest of this paper is structured as follows. In Section 2 we define our system model, some key properties of the model, and then define anonymity in the context of the proposed model. Section 3 identifies three different models of billing commonly found in cloud storage services, followed by an overview of relevant payment methods in Section 4. Section 5 ties the previous three sections together and investigates potential solutions for providing anonymity for clients of cloud storage services. Based on these findings, Section 6 presents avenues for future work and, finally, Section 7 concludes this paper.

2 System Model and Anonymity Definitions

In this section, we first define the system model that covers the relevant aspects for the discussion in this paper. After defining the model, we briefly discuss several general types of cloud storage and then define anonymity for clients of a cloud provider in the context of the introduced model.

2.1 Modelling a Cloud Storage Service

Let CP be a cloud provider who offers a public cloud storage service. Let $C = \{c_1, \ldots, c_m\}$ be the set of clients who store, read, write and delete data, and assume that c_i is the identifier of client i. We assume that there exists a one-to-one relation between the set of clients C and the set of natural persons, i.e., the natural person corresponding to client i can be interpreted to be c_i. We denote the amount of stored data, associated with a client c_i at a specific point in time, by s_i. Similarly, we denote the amount of bandwidth consumed by client c_i at a specific point in time as b_i.

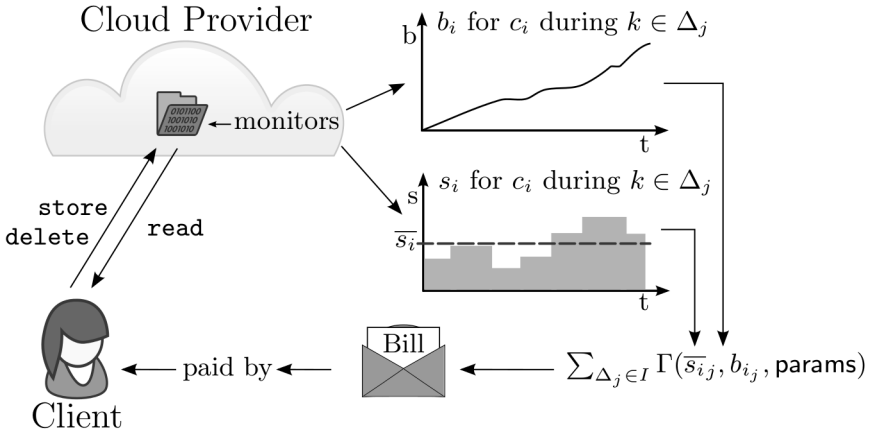

Figure 1: The system model where a client performs operations to `store`, `delete` and `read` data. The data is stored at a cloud provider that monitors each user's storage and bandwidth usage over time for each time interval. This is later used to generate a bill, for one or more time intervals, that the client has to pay.

Clients can perform `store` operations for a data object D of size d identified by id with the CP represented by an element $(id, c_i, d) \in \{0,1\}^* \times C \times \mathbb{N}$ and `delete` operations represented by an element $(id, c_i, d) \in \{0,1\}^* \times C \times \mathbb{Z} \setminus \mathbb{N}_0$. The meaning of these operations should be obvious and a `store` operation for an existing id can be interpreted as a write

operation. We note that in practice, cloud storage services usually offer key-value stores to clients, meaning that a `store` operation storing a data object D will return an identifier id for the stored data object in order to be able to retrieve it later on. A `read` or `delete` operation then amounts to providing id to the CP along with a corresponding flag indicating the operation to be performed. When data is read or stored, the client c_i consumes bandwidth and the total consumed bandwidth b_i (by the client c_i) is incremented accordingly. We consider the bandwidth used when deleting data negligible.

The time is discretised into intervals $\Delta_1, \ldots, \Delta_k$ where every interval $\Delta_i \in [0, t]$ has some length. For instance, one may think of 12 intervals of length as one month each and thus representing a year in total. So at the end of interval Δ_j client c_i may consume an amount of s_i storage units u_s (for instance, the unit u_s may be kB) and has consumed b_i units u_b of bandwidth (for instance, the unit u_b may be kb). Since the amount of storage used by a client over a time interval may fluctuate, for instance due to a temporary file being stored and then subsequently removed within the time interval, the *average* used storage during an interval is of interest[2]. We denote the average storage used by a client c_i during the current time interval as $\overline{s_i}$.

Next, we define p_s as the price of one unit of storage and p_b as the price of one unit of consumed bandwidth within an interval Δ_j. These prices allow us to specify the bill B_i for a client c_i for a subset of intervals $I \subseteq \{\Delta_1, \ldots, \Delta_k\}$. The bill B_i is defined as the sum of the bills of the corresponding intervals $B_i = \sum_{\Delta_j \in I}(p_{s_j} \cdot \overline{s_{i_j}} + p_{b_j} \cdot b_{i_j})$ weighted by the respective interval prices, where $\overline{s_i}$ is the average used storage and b_i the consumed bandwidth in each of the time intervals. Note that this is a very simple pricing policy, i.e., every unit within an interval is equally priced. However, one may define far more complex pricing policies. For instance, Amazon S3 charges for standard storage in the EU region (Ireland) for the first TB/month $\$0.0300$ per GB, for the next 49 TB/month $\$0.0295$ per GB, for the next 450 TB/month $\$0.0290$ per GB, etc[3]. A similar pricing model is in place for the consumed bandwidth. Furthermore, each operation has a very small fixed cost. In general, we can define some (public) pricing function $\Gamma(\overline{s_i}, b_i, \mathsf{params})$ whereas params represents a description of the pricing model and the bill is computed as $B_i = \sum_{\Delta_j \in I} \Gamma(\overline{s_{i_j}}, b_{i_j}, \mathsf{params})$ to cover a wider variety of pricing models. Figure 1 depicts this model. A client c_i is expected to pay by some means (cf. Section 4) for the bill B_i (cf. Section 3). This means that the client is required to pay real money equivalents for B_i to the CP and then B_i is set to zero. We denote the payment of a client c_i for a bill B_i by $P(c_i, B_i)$.

2.2 Types of Cloud Storage

Private customers are likely to use a cloud storage service as a simple backup infrastructure and/or as a designated place to store and share data, while enterprises are more likely to use it as a full-fledged virtual file system. The latter means that there are multiple users working in parallel and (extensive) access control mechanisms are required. However, we note that in the latter case it is likely that issues like access control are handled by a proxy running in the enterprises network perimeter and are not managed by the cloud storage service [72]. This means that from the point of view of a cloud provider an entire enterprise looks like a single client, i.e., the proxy that performs the operations on behalf of clients within the enterprise. Below, we briefly present meaningful types for cloud storage usage.

[2] Amazon S3 for instance uses this model of average consumption: `https://aws.amazon.com/s3/faqs/#How_much_does_Amazon_S3_cost`, accessed 2015-02-16.

[3] Amazon pricing in February 2015, `https://aws.amazon.com/s3/pricing/`.

Key to this classification are the following issues:

Single- vs. multi-reader: Only a single client (the data owner) will read data objects from the cloud storage that have previously been stored there. In the multi-reader setting, the data owner will share stored data objects (usually not duplicated) with other clients such that they can be read by them.

Single- vs. multi-writer: Only the data owner will write data objects (store, update) to its cloud storage space. In the multi-writer setting other clients are allowed to (over)write data at the data owner's storage space.

Thereby, we assume that such read or write (`store` in our model) operations are performed using the respective client's identity, e.g., if client c_i stores a data object with id_k in a multi-reader (multi-writer) setting and another client c_j performs a read (write) to id_k, then this client will perform the read (write) operation as c_j.

Typically, a cloud storage service may either be single-writer and single-reader (SW/SR), SW/MR or MW/MR. In the latter two types, we are not concerned with access control issues, i.e., how it is guaranteed that only privileged readers or writers can perform operations, but are only concerned with identity information accessible to CP, which allows behavioural tracking.

An issue that comes up in the latter two types is the question who should pay for operations conducted by clients that are different from the data owner – the owner or every client individually. For instance, in Amazon S3 by default, the owner of a bucket (a collection of data) pays all the costs associated with that bucket (for storage, data transfer, and requests). Besides, Amazon S3 lets a bucket owner designate a bucket as a so called *requester pays bucket*.[4] This means that each requester who accesses that particular bucket pays for their own data transfer and request costs. The bucket owner, however, still pays the storage costs. Clearly, the approach used (and they may be used in parallel) influences the billing.

2.3 Anonymity, Pseudonymity and Unlinkability

Anonymity as defined in [56] means, *that an attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set*, i.e., within the set of all potential subjects. In some settings such as anonymous communication channels, anonymity has been formalized by means of entropy, i.e., by a measure of an attacker's uncertainty [31, 64], whereas we need to stick with a more informal definition here (as discussed below).

Typically, in an anonymous setting, users and their actions can be connected to some kind of pseudonym (which does not provide an obvious link to the natural identity of the user). Depending on the type of pseudonym (cf. [56]) the anonymity guarantees of a system may be stronger or weaker. For instance, if a single pseudonym is used by one user during the entire lifetime of a system, then all actions of a single user can be *linked* to this pseudonym. Although this may not allow to identify the respective user, such a profile contains lots of information which may allow to identify the user, e.g., when given additional side channel information. Strongest anonymity guarantees are provided if a user uses distinct pseudonyms for every action (so called transaction pseudonyms in [56]) and thus all actions of a single user are *unlinkable* by pseudonym (cf. [74] for a generalisation of the entropy based anonymity metric to unlinkability).

[4] `http://docs.aws.amazon.com/AmazonDevPay/latest/DevPayDeveloperGuide/S3RequesterPays.html`, accessed 2015-02-16.

So, now why do we stick with somewhat informal definitions? Essentially, the problem is that we want to cope with a quite wide range of different payment and billing techniques (and their combinations). Furthermore, we cover billing, which in some cases *require* to link all actions within time intervals to a pseudonym. Similar, we do not differentiate between the SW/SR, SW/MR and MW/MW setting. Clearly, in the first case (SW/SR) it is obvious that all actions can be linked to the users pseudonym. However, and this is important, even if all actions within time intervals can be linked to one pseudonym, anonymous payment can still preserve the anonymity of the user. Consequently, we consider in our model all users actions, the billing and the payment and only say that such a system is anonymous, if for all these tasks the cloud provider cannot sufficiently identify the user within a set of all users with the information in our model.

2.4 Anonymity Definitions

When interacting with a set of clients C, the CP is able to collect information about the storage access and consumption behaviour of all clients in C. We denote this information as the *view* of the CP. This view includes information about the operations conducted by clients, the billing information available to the CP and the respective payment operations conducted by the clients and is the basis for our anonymity definitions.

Definition 1 (View of the CP)**.** A view V_{Δ_i} of the CP for an interval Δ_i and index sets I_{id} and I_d for data item identifiers and data item sizes respectively is defined as the collection of the subsequent information:

- **Operations:** Sequences
  $\sigma = ((id_{i_1}, c_1, d_{j_1}), \ldots, (id_{i_k}, c_1, d_{j_k}), \ldots, (id_{i'_1}, c_m, d_{j'_1}), \ldots, (id_{i'_\ell}, c_m, d_{j'_\ell}))$
  $\omega = ((id_{i_p}, c_1, d_{j_p}), \ldots, (id_{i_q}, c_1, d_{j_q}), \ldots, (id_{i'_r}, c_m, d_{j'_r}), \ldots, (id_{i'_s}, c_m, d_{j'_s}))$ of `store` (σ) and `delete` (ω) operations for $i, i' \in I_{id}$, $j, j' \in I_d$ respectively, together with a sequence $\rho = ((id_{i_t}, c_1), \ldots, (id_{i_u}, c_1), \ldots, (id_{i'_v}, c_m), \ldots, (id_{i'_w}, c_m))$ for $i, i' \in I_{id}$ of `read` (ρ) operations, conducted by clients in C.
- **Bills:** A set of bills $\mathcal{B} = \{(c_1, B_1), \ldots, (c_m, B_m)\}$ of all the clients in C.
- **Payments:** A sequence of payments $\pi = (P(c_1, B_1), \ldots, P(c_m, B_m))$ of all the clients in C for the respective bills.

Note that σ, ω, ρ and π can be considered as sequences ordered by their timestamps and we can implicitly assume that CP stores a timestamp for every element of these sequences. However, we do not explicitly include this information in these sequences for brevity.

The above defined view of the CP contains all the relevant users' behavioural information as well as billing and payment information. In order to be able to restrict ourselves to the information contained in the view, we assume that all the clients can communicate with CP via a (perfectly) anonymous channel. In practice, this can be approximated by using anonymous communication networks such as Tor [32], although such services are not immune to various types of attacks [45]. Otherwise, the communication channel could already reveal the client's unique identity and would render additional measures towards anonymous cloud storage meaningless.

Firstly, we make some observations and elaborate on the semantics of parts of the information in the view for different types of cloud storage services from Section 2.2. Therefore, in Table 1, we present on overview of which client identities for a single data object

id_k are included in the sequences σ, ω and ρ of view V_{Δ_ℓ} where c_i is the data owner and $c_i \in C' \subseteq C$. Assume that CP stores all views and maintains such a history of the lat-

Table 1: Influence of types of cloud storage on semantics of information in view V_{Δ_ℓ} for some specific data item id_k owned by c_i.

	σ (store)	ω (delete)	ρ (read)
SW/SR	c_i	c_i	c_i
SW/MR	c_i	c_i	C'
MW/MR	C'	C'	C'

est m views $\mathbb{H} = (V_{\Delta_\ell})_{\ell=1}^{m}$, and that in the MW/MR setting, writers that differ from the owner can only write/delete existing data objects (which is reasonable)[5]. However, then it is immediate that the information CP obtains from the sequences σ, ω and ρ in the SW/SR, SW/MR and MW/MR cloud storage setting are equivalent from an anonymity perspective. In particular, the owner of every data object id_k can also be easily determined in the MW/MR setting: simply go through $\mathbb{H}$ and since all the sequences in every view are ordered by the occurrence of their operations, take the first occurrence of id_k in σ and c_j in this tuple is the owner. Consequently, from an anonymity perspective we do not need to distinguish between the three types in Table 1.

Now, we are ready to state our anonymity definitions.

Definition 2 (Anonymously accessible cloud storage)**.** A cloud storage (CS) service is anonymously accessible iff all tuples in σ and ω are of the form $(id, ?, d)$ and all tuples in ρ are of the form $(id, ?)$ where $'?'$ means that the CP cannot sufficiently determine the natural person linked to the identity of the client within the set of all natural persons of all the clients of the CP[6].

Now, an *anonymously accessible cloud storage* could quite easily be realized using existing cryptographic techniques which support some kind of anonymous client authentication (neglecting access control, which however can also be realized anonymously as already mentioned in the related work section). However, so far we have not considered the billing and payment information in the view, which makes this task more complicated, since every client should only get billed for what it consumed and the CP should be sure to get paid correctly.

Definition 3 (Short-term anonymous cloud storage)**.** A CS service is anonymous iff **for each single** time interval Δ_ℓ all tuples in σ and ω are of the form $(id, ?, d)$, all tuples in ρ are of the form $(id, ?)$, all tuples in $\mathcal{B}$ are of the form $(?, B)$ and all tuples in π are of the form $P(?, B)$ where $'?'$ means that the CP cannot sufficiently determine the natural person linked to the identity of the client within the set of all natural persons of all the clients of the CP.

The above definition covers anonymity in the short term, i.e., within one (short) time interval. For achieving anonymity, this assumes that there is no *link* between time intervals, such that in essence some kind of *reset* occurs after each interval. However, in a realistic setting for cloud storage, data objects will reside at CP for more then one time interval,

[5] In a setting where the CP identifies users this may not be a problem, but if CP cannot, dishonest clients could flood the owner's storage space and produce high costs without being accountable.

[6] This definition is based on the anonymity definition by Pfitzmann and Hansen [56].

which leads to time intervals being linked by the data objects. This makes anonymity in the long term, as defined in Definition 4, much harder to achieve.

Definition 4 (Long-term anonymous cloud storage)**.** A CS service is anonymous iff **for all** time intervals all tuples in σ and ω are of the form $(id, ?, d)$, all tuples in ρ are of the form $(id, ?)$, all tuples in $\mathcal{B}$ are of the form $(?, B)$ and all tuples in π are of the form $P(?, B)$ where $'?'$ means that the CP cannot sufficiently determine the natural person linked to the identity of the client within the set of all natural persons of all the clients of the CP.

Now, we need to define how CP behaves. We consider the adversarial goal of the CP to be to identify *who* is storing *what* data at the CP. The CP attempts this identification while being *passive* (honest-but-curious), i.e., the CP does not divert from the specified protocols but attempts to deduce as much as possible from the available information in the views.

3 Types of Billing

We identify three different types of billing for cloud storage services. It is important to note that billing must not be confused with payment, as we consider a bill to be a representation of *what* a client has to pay for (with different levels of granularity), and, payment means *how a bill is settled* with (real world) money equivalents. We distinguish between three types of billing, ranging from very fine-grained billing closely coupled to immediate payment to coarse-grained billing relatively decoupled from immediate payment. The three different billing types are pay-as-you-go, subscription with a periodical aggregated bill, and subscription with a fixed periodical cost, i.e., flat rate.

3.1 Pay-As-You-Go

By pay-as-you-go billing we mean the case where $|I| = 1$ and $\Delta_i = 0$ for all i, i.e., intervals are of length zero, and, consequently every `store` (and potentially `read`) operation produces a single (typically independent) "bill". This bill presumably also contains any associated bandwidth costs, as part of the cost of storing or accessing data, but the primary variable that influences the amount to pay for an operation is the *size* d of the stored or read data. Furthermore, it is understood not only in the sense that the billing of storage space is fine-grained but also that the bill produced by every `store` operation is settled immediately, e.g. by spending credits. This typically means that a payment is required prior to an operation to prove the liquidity of the client, e.g., in the form of purchasing credits or some form of token. In this setting, while the CP does not need to keep track of any information for the sake of billing, each `store` (and potentially also each `read`) operation requires clients to pay. Consequently, the CP is not required to link different billings (operations) of a client together.

Applying this type of billing does per se not allow for complex pricing policies, only for fixed priced units. When tracking all single billings of a client during one larger "meta-interval", then a CP may provide some discount to the client within the meta-interval for future transactions. However, then it seems more profitable for the client to use aggregated billing as discussed next.

3.2 Aggregated

Aggregated billing is the state-of-the-art billing model for cloud storage services, such as Amazon S3. Here, each client c_i is registered with the CP and every `store`, `read`, and `delete` operation updates the values of s_i and b_i, with a typical interval size of one month. At the end of an interval, i.e., monthly, an aggregated bill B is computed using the pricing function based on the average used storage $\overline{s_i}$, and the consumed bandwidth b_i. The client is then required to pay the bill by some means. Here, the client can take advantage of many different types of discounts. In this setting, all operations and all data belonging to one client need to be linked together to form one bill at the end of one or more time intervals. Presumably, unless steps are taken to directly address this issue, the CP can link stored data (and thus all associated billing information) between time intervals. As a consequence of aggregated billing, the CP needs to keep detailed behavioural profiles of each client. When a bill is settled, there needs to be (per definition) a link to at least one instance of payment (the payment that settled the bill).

3.3 Flat Rate

In this billing model, the CP requires very coarse-grained up to no real billing information. Typically the CP provides a fixed amount of storage for free or may charge some fixed fee during an interval. A client may consume the storage space until this threshold is reached. There is no fine-grained billing, meaning that clients going above this threshold may be blocked or required to pay additional fees. The client is not reimbursed by the CP if the storage space is not fully utilised by the client. The bare minimum a client needs to convince the CP of are: (i) that the client is allowed to use this service in the current time interval, i.e., the client has paid, and (ii) the client is not over any threshold. Thereby, the CP may keep track of all information resulting from operations as in the case of aggregated billing, if no measures are employed to reduce/eliminate this information.

Table 2: Common payment methods in current commercial cloud storage services.

	Credit Card	PayPal	Bitcoin
ADrive (`http://www.adrive.com`)	✓	✓	–
Amazon S3 (`http://aws.amazon.com/s3/`)	✓	–	–
DataShell (`http://datashell.co.uk/`)	–	✓	✓
Dropbox (`http://www.dropbox.com`)	✓	✓	–
GoAruna (`http://goaruna.com`)	✓	–	✓
Google (`http://cloud.google.com/storage`)	✓	–	–
Wuala (`http://www.wuala.com`)	–	✓	–

4 Types of Payment

Clients may settle their bills at the CP by several means. We generally differentiate between traditional payment, where the payee can determine the identity of the payer, and anonymous payment, where the payer is anonymous towards the payee. Before we start the presentation of the various types of payments, in Table 2, we present the state of the art and in particular a selection of commercial cloud storage providers along with currently

supported payment methods. Note that this is not intended to be a selection of the most popular cloud storage services and some may also use each other (e.g., Dropbox is built upon Amazon S3), but serves to indicate supported payment methods.

4.1 Traditional (Non-Anonymous) Payment

In the context of cloud (storage) services the common payment methods are non-anonymous, which connect consumption profiles (and bills) of clients to PII via the payment process.

4.1.1 Credit Card & PayPal

Typically, client profiles are directly linked to credit cards and at the end of every interval, the CP charges the credit card. Other systems, for instance PayPal[7] or similar payment processors, hide the identity of the payer towards the CP. This does not constitute what we consider to be anonymous payment, since PayPal is in a position to easily identify any payer should it choose or be forced to. Furthermore, payment operations can easily be linked and users will typically not use such payment processors with the requirement of being anonymous in mind, which will make identification typically straightforward.

4.1.2 Bitcoin

Bitcoin is a peer-to-peer electronic currency system with no central authority such as a bank or issuer [53]. Transactions take place between public keys where ownership of a private key proves ownership of currency. All confirmed transactions that take place are grouped into *blocks* that form a *block chain*. The block chain is the publicly agreed upon spending history of every transaction that has ever occurred. Blocks are appended to the block chain by *proofs of work* based on finding a hash of one or more broadcasted transactions together with a variable nonce that provides an output with a desired number of prefixed zeroes. Bitcoin was not designed to provide anonymity [53, 61] and current implementations do not provide anonymity [2]. How unlinkable Bitcoin transactions are to users of Bitcoin, i.e., if the payers are anonymous or not, ultimately depends on how the Bitcoins were *acquired*. If a user acquires Bitcoins through a reseller by using traditional payment methods, such as a credit card, then the ability to link the Bitcoins to the identity of the user depends on the reseller. If users generate their own Bitcoins by performing proofs of work and assign the resulting rewarded Bitcoins to a newly created public key, then using these Bitcoins to pay for a service (such as a cloud storage service) leaks little more information other than the fact that the transaction took place.

Recent work on extensions to the Bitcoin ecosystem that do not modify the original system but additional add an support for fully anonymous transactions, in the form of Zerocoin [52] or its adoption to the elliptic curve setting called Pinocchio Coin [29], is an interesting direction, but not yet widely used. The same holds for quite recent proposals to use Bitcoin as a building block for anonymous transactions [7, 10].

4.2 Anonymous Payment

With anonymous payment we refer to the case when the payer that is paying the recipient is anonymous, i.e., the payee cannot determine the identity of the payer. This should ideally

[7] `https://www.paypal.com`, accessed 2015-02-16.

be the case even when the payee colludes with other entities in the system, such as an issuer of the currency or a bank that plays a central role in the system. In other words, we are primarily interested in anonymous payment systems where the end result is that a Trusted Third Party (TTP) that handles payment is *simulated* through cryptography by the anonymous payment system (in contrast to systems like PayPal).

4.2.1 Anonymous E-Cash

In E-Cash systems [24], users can withdraw a number of electronic coins from a bank using a withdrawal protocol and then spend (pay) these coins at merchants. Merchants can then exchange electronic coins for regular cash on their bank account using a deposit protocol with the bank. E-cash should provide users with anonymity towards both the bank and the merchant during a purchase, but spending multiple copies of one coin (double-spending) should be prohibited. Double spending may only be detectable, as in on-line systems [24], where for every transaction between users and merchants the bank is in contact with the merchant. Additionally, double-spending may also involve revealing the identity of the cheater, as in off-line systems [11], where the bank is not directly involved in transactions. This anonymity revocation can either be performed by a TTP who can determine the identity of a coin's holder in any case, or by cryptographic means, i.e., identification is only possible in case of a double-spending event. Over the years, numerous variants have been proposed, such as transferable, endorsed, compact and divisible E-Cash systems. Most interesting in the cloud storage setting seem to be compact [14] as well as divisible E-Cash [18] systems. Compact E-Cash [14, 4] allows a user to withdraw several coins (a wallet) in a single transaction, that is cheaper than withdrawing k single coins. A more sophisticated approach is the concept of divisible E-Cash [5, 18], where such schemes firstly allow a user to withdraw a wallet of value 2^ℓ in a single withdraw protocol. Secondly, spending a value 2^m for $m \leq \ell$ can be realised more efficient than repeating the spending 2^m times. Furthermore, anonymity as well as unlinkability of different withdrawals are guaranteed.

4.2.2 Anonymous Micropayments

In contrast to standard electronic cash transactions, micropayments are intended for only a very small amount of money to be payed within a transactions. There are numerous cryptographic systems which have been proposed in the past, such as PayWord and MicroMint [62], PayTree [48], etc. (cf. [27] for more approaches).

Only quite recently, anonymous micropayments focusing on users privacy have been introduced in context of networked services (and in particular anonymous communication networks) [3, 27]. In the most recent work [21], the authors propose a set of anonymous micropayment mechanisms where users can make untraceable, anonymous micropayments and several micropayments can be aggregated and cashed together.

Since traditional E-Cash can often becomes too expensive, in both cost and execution times, for arbitrary small payments, anonymous micropayments represent an alternative, since these approaches are typically orders of magnitudes more efficient.

4.2.3 Anonymous Prepaid Mechanisms

By prepaid mechanisms we mean that clients purchase credits in advance for use of the cloud storage service. This type of payment is well known from pre-paid mobile phone services and is typically non-anonymous, i.e., is linked to some credit card or bank account

of the client. Moreover, this concept is often used in context of digital rights management (DRM) [46] or in electronic (mulit-) coupon systems [19, 25].

Recently, anonymous prepaid mechanisms tailored to the use within cloud services have been proposed [66, 57]. The main idea behind such schemes is that clients are able to purchase a contingent of credits — represented as a single compact token whose size is independent from the number of credits — and consumes credits from this token for storage space from some reseller. While spending credits for resources at the CP, the CP does not learn anything about the resource consumption behaviour of clients. Users thus can anonymously and unlinkably consume repeatedly an arbitrary number of their credits from the token as long as there are still enough credits in the token available.

5 Evaluation of Feasibility of Anonymous Cloud Storage

In this section we tie together the system model from Section 2, the three different billing models from Section 3, and the two different payment models from Section 4. The two payment models are the traditional model, where the payee can identify who the payers are, and the anonymous model, where the payer is anonymous towards the payee. We start by looking at the traditional payment setting. Next, we look at different privacy-enhancing solutions for providing anonymity for clients while still using traditional payment. Last, but not least, we discuss the impact of clients being able to pay anonymously.

5.1 The Traditional Setting

The traditional setting uses the traditional payment model where the CP can identify the client that is paying for a particular bill. In this setting, the storage service provided by the CP has not been designed with providing anonymity for clients in mind, i.e., π in the CP's view always contains the clients' identities. This setting represents the vast majority of cloud storage services available today, and the type of billing makes little difference in terms of anonymity:

Pay-As-You-Go: Each `store` and `read` operation in σ, ω and ρ will contain the identity of the client together with a direct payment made by the client to immediately settle the "bill".

Aggregated: All operations in σ, ω and ρ by a client are linked to the client to allow generation of the bill B at the end of the time interval.

Flat Rate: For each time interval, presumably at the very start, the client has to pay to access the service. All operations in σ, ω and ρ identify the client such that the CP can be sure that the client has paid to use the service in the current time interval.

There is no anonymity for clients of the CP, regardless of the billing model, as long as traditional payment is used in the traditional setting and the service was not designed with privacy in mind. The view of the CP contains the identity of the client in all operations, and these operations are linked together with the payment that settles a bill, where the payment also reveals the identity of the client.

There are, however, a number of different existing privacy-enhancing solutions that may enable clients to remain anonymous towards the CP, while still using traditional payment. We identify the following three key groups: anonymous credentials, multi-user oblivious RAM (ORAM), and the use of trusted third parties.

5.1.1 Anonymous Credentials

Anonymous credentials enable clients to be authenticated and authorised anonymously at a CP. Multi-show credentials [15, 16, 17, 20, 39] allow showing the same credentials multiple times while being unlinkable, whereas one-show credentials [12, 6] allow only a single showing in an unlinkable fashion.

The use of anonymous credentials for short-term anonymous cloud storage, when combined with traditional billing, is limited to the case of flat rate billing. The CP may issue anonymous credentials upon payment to clients such that clients may convince the CP of the fact that they have a valid credential without revealing any other information, as is done in [58]. This would allow the user to be anonymous in the short term, as defined in Definition 3. Since the client pays *before* any data is uploaded, the CP has no a-priori information about the client and the payment is decoupled from the client's identity (pseudonym, potentially provided by the anonymous credential as in [58]) at the CP. If the client continued to, per time interval, purchase anonymous credentials from the CP to access its data at the CP, there would however be stored data (and associated information) belonging to the client at the CP from the past. Over time, this would allow the CP to perform an *intersection attack* [60] on the sets of clients of the CP for each time interval, found in the CP's history of all views $\mathbb{H} = (V_{\Delta_\ell})_{\ell=1}^{m}$, ultimately linking clients to their data given enough time. Therefore, long-term anonymity is not possible using anonymous credentials alone, regardless of the type of billing.

Aggregated billing has one major downside when compared to flat rate billing: the bill is likely unique for each client. This results in usage (σ, ω and ρ) for information used to generate a bill B for a likely unique amount that is paid by a client using the client's natural identity. This makes even short-term anonymity unlikely for aggregated billing.

In pay-as-you-go billing, each operation can be seen as a time interval of its own. This means that, while short-term anonymity for one operation may be achievable, long-term anonymity is not possible for reasons previously discussed. Another notable restriction for pay-as-you-go is that the amount one pays for has to be generic and not too fine-grained, otherwise the amount paid will be unique and thus easily linked to the data.

5.1.2 Multi-User ORAM

ORAM [36, 73] is a well known technique to obliviously read and write data items from a remote storage[8]. An ORAM may be set up by the CP for some defined storage space for a set of clients, which is initialised with "bogus data" and clients can then store, delete and read data items obliviously without the necessity to reveal their identity and which item they are accessing. Since ORAM in essence hides *what* data is stored, deleted and read it is clear that clients can be both short- and long-term anonymous, simply because stored data cannot be observed to begin with. This is, of course, assuming that a multi-user ORAM [34, 38, 50, 51] is used where multiple clients use the same ORAM. The list of clients for a ORAM then becomes the anonymity set for the clients, where those clients are all potential owners of any piece of data stored in the ORAM.

When it comes to types of billing, flat rate and pay-as-you-go billing are natural for a multi-user ORAM. Aggregated billing on the other hand suffers from the fact that all store, delete and read operations on an ORAM is costly in terms of overhead when compared

[8]Note, that we are not interested in partial solutions where only reads are oblivious, which can for instance be achieved by means of Private Information Retrieval (PIR) [28].

to regular cloud storage. Since the ORAM is initialised with "bogus data" that is indistinguishable from "real data", clients have to pay for the full ORAM (or a slice of it, depending on how many clients are users of the same ORAM) from the beginning. While conceptually this is acceptable, having to convey to clients the implications of ORAM may make aggregated billing less compelling. In essence, what can be aggregated is the number of accesses, not amount of data stored or transferred.

5.1.3 Trusted Third Parties

Instead of having the CP accept traditional payments directly from clients, one approach is to introduce a trusted *payment processor* (PP) that acts as a trusted third party (TTP). Clients pay to the PP who in turn forwards the money to the CP without revealing the identity of the client. Another similar approach is to introduce a trusted *cloud reseller* (CR). The CR buys tokens from the CP that the CR later resells to clients. A client can then pay the CP with tokens to settle a bill without revealing his or her identity to the CP. The CP may offer any billing model, since all operations in σ, ω and ρ can be tied to a pseudonym. Each bill B for a client in $\mathcal{B}$ can be associated to the pseudonym as well. At payment, the information about the natural identity of the client is only known by the PP or CR, not the CP. As long as the PP or CR is trusted, the user can have both long and short-term anonymity since the number of time intervals has no impact on the amount of identifying information available to the CP in our model.

5.1.4 Remarks

For flat rate billing, anonymous credentials can be used to achieve short-term anonymous cloud storage as shown in [58]. For pay-as-you-go and aggregated billing, anonymous credentials alone are insufficient due to the billing models short time intervals and the required granularity of the bill, respectively. Multi-user ORAM can provide long-term anonymous cloud storage for all types of billing. Flat rate and pay-as-you-go billing is a natural fit, whereas aggregated billing becomes convoluted due to the large overheads associated with ORAM. Introducing a PP or CR is only a solution to the anonymity problem when traditional payments are used, provided that they are trusted. Expanding the adversary model to also consider outside threats, such as the threat posed by legal pressures put on the PP, CR and CP, invalidates these TTP solutions because none of the entities can be trusted [69].

5.2 Using Anonymous Payment

With an anonymous payment scheme, the client c_i can simply use a pseudonym c_i' at CP such that all parts of the view of the CP only contains the pseudonym and not the natural identity of the client. The client is then anonymous in the long term, as defined in Definition 4. This is because in essence the goal of anonymous payment is to simulate, with the help of cryptography, a TTP that manages payment. This scenario was discussed above in the traditional payment setting. The difference between the different types of billing is then simply the size of the profile associated to the pseudonyms of clients that the CP has to maintain for billing purposes. The more information in the profile, the more vulnerable the client is to being identified due to some side channel information outside of our model.

5.3 Overview

In Table 3, we provide a compact overview of the feasibility to realise anonymous cloud storage, short term as defined in Definition 3 and long term as defined in Definition 4, using different types of billing and payment. Note that, when using traditional payment,

Table 3: An overview of how the combination of different types of billing and payment allow anonymous cloud storage with respect to Definitions 3 and 4. $\times$ indicates that anonymity is not possible, $\approx$ indicates that anonymity can be realised with some conditions or assumptions, and ✓ denotes that anonymity is possible.

	Short-Term Anonymity	Long-Term Anonymity
Traditional Payment		
Pay-As-You-Go	$\times$	$\times$
Aggregated Billing	$\times$	$\times$
Flat Rate	$\times$	$\times$
P-E: Anonymous Credentials		
Pay-As-You-Go	$\approx$	$\times$
Aggregated Billing	$\times$	$\times$
Flat Rate	✓	$\times$
P-E: Multi-User ORAM		
Pay-As-You-Go	✓	✓
Aggregated Billing	$\approx$	$\approx$
Flat Rate	✓	✓
P-E: Trusted Third Party		
Pay-As-You-Go	$\approx$	$\approx$
Aggregated Billing	$\approx$	$\approx$
Flat Rate	$\approx$	$\approx$
Anonymous Payment		
Pay-As-You-Go	✓	✓
Aggregated Billing	✓	✓
Flat Rate	✓	✓

long-term anonymous cloud storage is only possible with privacy-enhancing solutions that either result in significant overhead, in the case of multi-user ORAM, or by introducing trust[9], as in the case of using a TTP. For multi-user ORAM, the aggregated billing model becomes convoluted due to part of the meaning of aggregated billing becoming lost, since ORAM hides the size of the actual data stored, how much data is actually read etc. Anonymous credentials alone provide at best short-term anonymity, breaking down long term due to correlation attacks. Solutions based on anonymous payment, as previously mentioned, provide long-term anonymity regardless of the type of billing because they allow payment by simulating a TTP that handles payment. Depending on the type of anonymous payment, there is little to no extra trust needed, in comparison to actually fully TTPs.

[9] We note that solving a security or privacy problem by introducing extra trust in a single entity is arguably not a valid solution in many instances, hence the $\approx$ symbol in Table 3.

6 Avenues for Future Work

The overview from Section 5 provides us with at least two interesting avenues for future work: accurate privacy-preserving aggregated billing and addressing long-term anonymous cloud storage based on anonymous credentials.

For aggregated billing, interesting future work is to realise aggregated billing without the CP being able to link single actions together (not even to a per interval pseudonym), while being sure that the aggregate at the end of the interval reflects the correct consumption behaviour of a client. This is somewhat related to the work in [30] in context of smart metering. One may call this *oblivious aggregated billing* and one can envision the use of existing cryptographic building blocks to solve this problem: For instance, let $(C_1, \ldots, C_n)$ be a sequence of homomorphic commitments where C_i represents s_i of client c_i for some given time interval Δ_j (initially s_i is a commitment to value 0). When performing a write operation for data object of size d, c_i creates commitments $C'_1, \ldots, C'_n$ where C'_i is a commitment to d and all others to 0. A user would then need to prove in zero-knowledge that 1 out of the n commitments contain value d and all other 0 and then the commitments are added homomorphically. The remaining problem to solve in this first construction is to guarantee, that client c_i is only allowed to add to his commitment C_i the value d, but not to cheat and add d to any other client's commitment. Unfortunately, it is not clear how to resolve this issue. However, there may be alternative (more efficient) ways to realize *oblivious aggregated billing*.

Another interesting direction for future work is to figure out how to do an efficient *verifiable shuffle* of all the data stored at the CP, such that the CP cannot link the data before and after the shuffling, while clients retain the ability to access their data. By performing a verifiable shuffle at the end of each time interval, a solution (such as anonymous credentials) that provides short-term anonymity as defined in Definition 3 could be modified to provide long-term anonymity as defined in Definition 4. We are not aware of any work on verifiable shuffles in the cloud storage setting together with anonymous credentials, but there is related work. In the cloud storage area, recent work by Stefanov and Shi [70] use a shuffle between two or more non-colluding cloud storage providers for ORAM, greatly decreasing the cost of running an ORAM in terms of client-cloud bandwidth. In other areas, solutions based on mix-nets [23], like that of Wikström [77], enable a verifiable shuffle of data such that nobody learns the correspondence between input and output (assuming a subset of mix servers is honest), where mix servers can prove that they performed the shuffle correctly. Such an approach could be used by multiple cloud providers to shuffle the data they store for their users in a verifiable way as long as some cloud providers remain honest, as in [70]. Another source of potential solutions is the classical area of mental poker, where shuffling a deck of cards without a trusted party is paramount [65]. Recent work, however, still shows significant overheads in this setting [44]. One major downside to an approach from this area is that the client might have to be online. With an online client, the foundation to improve upon is the naive solution of having the client download, transform, and then re-upload the data such that the CP could not link it to the old data for each time interval. The advantage of the approach based on mix servers, or that of [70], is that multiple cloud storage providers could collaborate to make the mixing transparent (yet still provable) to clients without direct interactions with clients. Despite the improvement in performance for ORAMs in [70], utilising shuffles together with anonymous credentials or other short-tern anonymity solutions seems like fruitful future work with potential for even better performance.

Moreover, it would be interesting to come up with a complete abstract model that captures

anonymous cloud storage and allows to formally analyse and quantify the anonymity provided by different approaches. However, in contrast to anonymous communication channels, which seem to allow easier abstraction and formal models allowing quantification of anonymity [31, 64] are available for quite a long time, this does not seem to be that easy for the subject at hand.

7 Conclusions

State-of-the-art commercial cloud storage services are not designed with the privacy of customers in mind. Although encrypting data prior to outsourcing it to a cloud storage provider guarantees data privacy, there is still a plethora of potentially privacy-critical information available to the cloud provider. In context of encryption, it should be noted that encrypting data prior to outsourcing requires some care in the choice of the used encryption scheme and the used security parameter. Encryption schemes (and choices of parameters) which are secure today may not be secure anymore in the (near) future and this could be problematic for highly sensitive data.

Since to date only non anonymous payment methods are supported, cloud providers can link any stored data object as well as every operation on data objects to the respective client. Cloud providers may use this "side channel information" to learn the access history, which reveals users' habits and privileges and this can be privacy intrusive [26].

Motivated by the lack of privacy-preserving measures in existing commercial cloud storage services, in this paper, we investigate the feasibility of so called *anonymous cloud storage* services. While various measures to realize access privacy have been studied in the past, until now, the role of privacy in context of billing and payment has remained unexplored. Our work leads us to the following conclusions:

- State-of-the-art commercial cloud storage services using traditional payment cannot achieve anonymous cloud storage as defined in this paper.
- Although short term anonymous cloud storage can be achieved without anonymous payment, long term anonymous cloud storage is desirable to achieve, since it represents the most important real world use case.
- By applying existing privacy-enhancing technologies, even when using traditional payment, short term anonymity can be achieved efficiently by the use of anonymous credentials (although the aggregated billing approach is problematic). Even long term anonymous cloud storage can be achieved by the use of mulit-user ORAM, but at unreasonably high costs.
- Using a TTP and traditional payment allows long- as well as short-term anonymous cloud storage. However, the unlikely existence of a truly trustworthy TTP in practice makes this approach unappealing [69].
- Oblivious aggregated billing and efficient verifiable shuffling of data are open challenges for future research. They would allow long term anonymous cloud storage even without anonymous payment.

For practical commercial anonymous cloud storage services, the lack of widely deployed and easy to use anonymous payment mechanisms remains a problem. A promising recent development within the Bitcoin ecosystem are Zerocoin variants [52, 29] and approaches

based on Bitcoin [7, 10] that add an support for fully anonymous transactions to Bitcoin transactions and seems to be an interesting direction. Besides, anonymous credentials systems are brought to practice [10] and research on how to realise long-term anonymous cloud storage using anonymous credentials appears to be another another fruitful direction.

Lastly, we would like to note that legal and regulatory hurdles may make any progress on the technicality of practical commercial anonymous cloud storage moot. For example, The European Payment Services Directive 2007/64/EC (PSD) requires in Article 39 that third-party payment service providers provide "*a reference enabling the payer and the payee to identify the payment transaction and the payer, where appropriate, and any information transferred with the payment transaction*". Any potential anonymous payment scheme including a payment service that falls under PSD risks violating the directive by being unable to provide necessary information, thus making it illegal in Europe.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable suggestions. Tobias Pulls has been funded by a Google research award on "Usable Privacy and Transparency Tools II". Daniel Slamanig has been supported by the Austrian Research Promotion Agency (FFG), grant agreement number 832145.

References

[1] Balamurugan Anandan, Chris Clifton, Wei Jiang, Mummoorthy Murugesan, Pedro Pastrana-Camacho, and Luo Si. *t*-Plausibility: Generalizing Words to Desensitize Text. *Transactions on Data Privacy*, 5(3):505–534, 2012.

[2] Elli Androulaki, Ghassan Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating User Privacy in Bitcoin. In *Financial Cryptography*, LNCS. Springer, 2013.

[3] Elli Androulaki, Mariana Raykova, Shreyas Srivatsan, Angelos Stavrou, and Steven M. Bellovin. PAR: Payment for Anonymous Routing. In *Privacy Enhancing Technologies*, volume 5134 of *LNCS*, pages 219–236. Springer, 2008.

[4] Man Ho Au, Willy Susilo, and Yi Mu. Practical Compact E-Cash. In *Australasian Conference on Information Security and Privacy*, volume 4586 of *LNCS*, pages 431–445. Springer, 2007.

[5] Man Ho Au, Willy Susilo, and Yi Mu. Practical Anonymous Divisible E-Cash from Bounded Accumulators. In *Financial Cryptography*, volume 5143 of *LNCS*, pages 287–301. Springer, 2008.

[6] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous Credentials Light. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*. ACM, 2013.

[7] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014*, pages 459–474. IEEE, 2014.

[8] Igor Bilogrevic, Julien Freudiger, Emiliano De Cristofaro, and Ersin Uzun. What's the Gist? Privacy-Preserving Aggregation of User Profiles. In *Computer Security - ESORICS 2014*, volume 8713 of *LNCS*, pages 128–145. Springer, 2014.

[9] Marina Blanton. Online subscriptions with anonymous access. In *2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008*, pages 217–227. ACM, 2008.

[10] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. In *Financial Cryptography*, volume 8437 of *LNCS*, pages 486–504. Springer, 2014.

[10] See, e.g., the ABC4Trust EU project, `https://abc4trust.eu/`, accessed 2015-02-16.

[11] Stefan Brands. Untraceable Off-line Cash in Wallets with Observers (Extended Abstract). In *CRYPTO*, volume 773 of *LNCS*, pages 302–318. Springer, 1993.

[12] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates*. MIT Press, 2000.

[13] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious Transfer with Access Control. In *2009 ACM SIGSAC Conference on Computer and Communications Security, CCS'09*, pages 131–140, 2009.

[14] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact E-Cash. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.

[15] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[16] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Security and Cryptography for Networks*, volume 2576 of *LNCS*, pages 268–289. Springer, 2002.

[17] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *CRYPTO*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[18] Sébastien Canard and Aline Gouget. Multiple Denominations in E-cash with Compact Transaction Data. In *Financial Cryptography*, volume 6052 of *LNCS*, pages 82–97. Springer, 2010.

[19] Sébastien Canard, Aline Gouget, and Emeline Hufschmitt. A Handy Multi-coupon System. In *Applied Cryptography and Network Security*, volume 3989 of *LNCS*, pages 66–81. Springer, 2006.

[20] Sébastien Canard and Roch Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In *ASIACCS*, pages 381–392. ACM, 2013.

[21] Bogdan Carbunar, Yao Chen, and Radu Sion. Tipping pennies? privately practical anonymous micropayments. *IEEE TIFS*, 7(5):1628–1637, 2012.

[22] Claude Castelluccia, Mohamed Ali Kâafar, and Minh-Dung Tran. Betrayed by Your Ads! - Reconstructing User Profiles from Targeted Ads. In *Privacy Enhancing Technologies*, volume 7384 of *LNCS*, pages 1–17. Springer, 2012.

[23] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[24] David Chaum. Blind Signatures for Untraceable Payments. In *CRYPTO*, pages 199–203, 1982.

[25] Liqun Chen, Alberto N. Escalante, Hans Löhr, Mark Manulis, and Ahmad-Reza Sadeghi. A Privacy-Protecting Multi-Coupon Scheme with Stronger Protection Against Splitting. In *Financial Cryptography*, volume 4886 of *LNCS*, pages 29–44. Springer, 2007.

[26] Yanpei Chen, Vern Paxson, and Randy H. Katz. What's New About Cloud Computing Security? Technical Report UCB/EECS-2010-5, UC Berkeley, 2010.

[27] Yao Chen, Radu Sion, and Bogdan Carbunar. XPay: Practical anonymous payments for Tor routing and other networked services. In *ACM Workshop on Privacy in the Electronic Society, WPES 2009*, pages 41–50. ACM, 2009.

[28] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *IEEE Annual Symposium on Foundations of Computer Science, FOCS 1995*, pages 41–50. IEEE, 1995.

[29] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio Coin: Building Zerocoin from a Succinct Pairing-based Proof System. In *PETShop 2013*, pages 27–30. ACM, 2013.

[30] George Danezis, Markulf Kohlweiss, and Alfredo Rial. Differentially Private Billing with Rebates. In *Information Hiding*, volume 6958 of *LNCS*, pages 148–162. Springer, 2011.

[31] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Privacy Enhancing Technologies*, volume 2482 of *LNCS*, pages 54–68. Springer, 2002.

[32] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security*, pages 303–320, 2004.

[33] Alan Dunn, Johnatan Katz, Michael Lee, Brent Waters, and Emmett Witchel. Anon-Pass: Practical Anonymous Subscriptions. In *2013 IEEE Symposium on Security and Privacy, SP 2013*. IEEE, 2013.

[34] Martin Franz, Peter Williams, Bogdan Carbunar, Stefan Katzenbeisser, Andreas Peter, Radu Sion, and Miroslava Sotáková. Oblivious Outsourced Storage with Delegation. In *Financial Cryptography*, volume 7035 of *LNCS*, pages 127–140, 2011.

[35] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *2007 IEEE Symposium on Security and Privacy, SP 2007*, pages 131–148. IEEE, 2007.

[36] Oded Goldreich and Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.

[37] Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Practical Oblivious Storage. In *ACM Conference on Data and Application Security and Privacy, CODASPY 2012*, pages 13–24. ACM, 2012.

[38] Michael T. Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Privacy-Preserving Group Data Access via Stateless Oblivious RAM Simulation. In *ACM-SIAM SODA*, pages 157–167, 2012.

[39] Christian Hanser and Daniel Slamanig. Structure-Preserving Signatures on Equivalence Classes and their Application to Anonymous Credentials. In *ASIACRYPT*, volume 8873 of *LNCS*. Springer, 2014. Full version: Cryptology ePrint Archive, Report 2014/705.

[40] Ryan Henry, Yizhou Huang, and Ian Goldberg. One (Block) Size Fits All: PIR and SPIR with Variable-Length Records via Multi-Block Queries. In *Annual Network and Distributed System Security Symposium, NDSS 2013*, 2013.

[41] Yizhou Huang and Ian Goldberg. Outsourced Private Information Retrieval with Pricing and Access Control. In *ACM Workshop on Privacy in the Electronic Society, WPES 2013*. ACM, 2013.

[42] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, and Peter-Paul de Wolf. *Statistical Disclosure Control*. Wiley, 2012.

[43] John Ioannidis, Sotiris Ioannidis, Angelos D. Keromytis, and Vassilis Prevelakis. Fileteller: Paying and Getting Paid for File Storage. In *Financial Cryptography*, volume 2357 of *LNCS*, pages 282–299. Springer, 2002.

[44] Tzer jen Wei. Communication efficient shuffle for mental poker protocols. *Inf. Sci.*, 181(22):5053–5066, 2011.

[45] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*, pages 337–348. ACM, 2013.

[46] Nakul Joshi and Ronald Petrlic. Towards practical privacy-preserving digital rights management for cloud computing. In *CCNC*, pages 265–270. IEEE, 2013.

[47] Taeho Jung, Xiang-Yang Li, Zhiguo Wan, and Meng Wan. Privacy preserving cloud data access with multi-authorities. In *INFOCOM*, pages 2625–2633. IEEE, 2013.

[48] Charanjit S. Jutla and Moti Yung. PayTree: "Amortized Signature" for Flexible Micro-Payments. In *USENIX Workshop on Electronic Commerce*, 1996.

[49] Seny Kamara and Kristin Lauter. Cryptographic Cloud Storage. In *Financial Cryptography Workshops*, volume 6054 of *LNCS*, pages 136–149. Springer, 2010.

[50] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Brief announcement: towards security and privacy for outsourced data in the multi-party setting. In *ACM Symposium on Principles of Distributed Computing, PODC '14*, pages 144–146, 2014.

[51] Matteo Maffei, Giulio Malavolta, Manuel Reinert, and Dominique Schröder. Privacy and Access

Control for Outsourced Personal Records. In *2015 IEEE Symposium on Security and Privacy, SP 2015*. IEEE, 2015.

[52] Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *2013 IEEE Symposium on Security and Privacy, SP 2013*, pages 397–411. IEEE, 2013.

[53] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009.

[54] Vinícius M. Pacheco and Ricardo Staciarini Puttini. SaaS Anonymous Cloud Service Consumption Structure. In *ICDCS Workshops*, pages 491–499. IEEE, 2012.

[55] Vinícius M. Pacheco and Ricardo Staciarini Puttini. Untraceable Anonymous Service Consumption in SaaS. In *CLOSER*, pages 96–101. SciTePress, 2012.

[56] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymyity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, August 2010.

[57] Martin Pirker, Daniel Slamanig, and Johannes Winter. Practical Privacy Preserving Cloud Resource-Payment for Constrained Clients. In *Privacy Enhancing Technologies*, volume 7384 of *LNCS*, pages 201–220. Springer, 2012.

[58] Tobias Pulls. Privacy-friendly cloud storage for the data track – an educational transparency tool. In *Nordic Conference on Secure IT Systems, NordSec 2012*, volume 7617 of *LNCS*. Springer, 2012.

[59] Mariana Raykova, Hang Zhao, and Steven M. Bellovin. Privacy Enhanced Access Control for Outsourced Data Sharing. In *Financial Cryptography*, volume 7397 of *LNCS*, 2012.

[60] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In *Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*. Springer, 2001.

[61] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and Privacy in Social Networks*, pages 197–223. Springer, 2011.

[62] Ronald L. Rivest and Adi Shamir. PayWord and MicroMint: Two Simple Micropayment Schemes. In *Security Protocols Workshop*, volume 1189 of *LNCS*, pages 69–87. Springer, 1996.

[63] Vyas Sekar and Petros Maniatis. Verifiable resource accounting for cloud computing services. In *ACM Cloud Computing Security Workshop, CCSW 2011*, pages 21–26. ACM, 2011.

[64] Andrei Serjantov and George Danezis. Towards an Information Theoretic Metric for Anonymity. In *Privacy Enhancing Technologies*, volume 2482 of *LNCS*, pages 41–53. Springer, 2002.

[65] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman. Mental Poker. Technical Report LCS/TR-125, MIT, 1979.

[66] Daniel Slamanig. Efficient Schemes for Anonymous yet Authorized and Bounded Use of Cloud Resources. In *Selected Areas in Cryptography*, volume 7118 of *LNCS*, pages 73–91, 2011.

[67] Daniel Slamanig. Dynamic Accumulator Based Discretionary Access Control for Outsourced Storage with Unlinkable Access. In *Financial Cryptography*, volume 7397 of *LNCS*, pages 215–222. Springer, 2012.

[68] Daniel Slamanig and Christian Hanser. On Cloud Storage and the Cloud of Clouds Approach. In *ICITST*, pages 649 – 655. IEEE, 2012.

[69] Christopher Soghoian. Caught in the cloud: Privacy, encryption, and government back doors in the Web 2.0 era. *J. on Telecomm. and High Tech. L.*, 8(2):359–424, Spring 2010.

[70] Emil Stefanov and Elaine Shi. Multi-Cloud Oblivious Storage. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*, pages 247–258. ACM, 2013.

[71] Emil Stefanov and Elaine Shi. ObliviStore: High Performance Oblivious Cloud Storage. In *2013 IEEE Symposium on Security and Privacy, SP 2013*, pages 253–267. IEEE, 2013.

[72] Emil Stefanov, Marten van Dijk, Ari Juels, and Alina Oprea. Iris: A Scalable Cloud File System with Efficient Integrity Checks. In *Annual Computer Security Applications Conference, ACSAC 2012*, pages 229–238. ACM, 2012.

[73] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13*. ACM, 2013.

[74] Sandra Steinbrecher and Stefan Köpsell. Modelling Unlinkability. In *Privacy Enhancing Technologies*, volume 2760 of *LNCS*, pages 32–47. Springer, 2003.

[75] Latanya Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[76] Matthew Wachs, Lianghong Xu, Arkady Kanevsky, and Gregory R. Ganger. Exertion-based Billing for Cloud Storage Access. In *HotCloud*. USENIX, 2011.

[77] Douglas Wikström. A commitment-consistent proof of a shuffle. *Cryptology ePrint Archive*, 2011:168, 2011.

[78] Saman Zarandioon, Danfeng (Daphne) Yao, and Vinod Ganapathy. K2C: Cryptographic Cloud Storage with Lazy Revocation and Anonymous Access. In *SecureComm*, volume 96 of *LNICST*, pages 59–76. Springer, 2011.