More Privacy for Cloud Users: Privacy-Preserving Resource Usage in the Cloud

Daniel Slamanig

Carinthia University of Applied Sciences, Primoschgasse 10, 9020 Klagenfurt, Austria.
d.slamanig@cuas.at

Abstract. Contrary to many other approaches to security and privacy in the cloud, we are interested in the problem of hiding behavioral information of users consuming their cloud resources, e.g. CPU time or storage space, from the cloud provider. More precisely, we are looking for solutions that allow users to purchase a contingent of resources from a cloud provider and to anonymously consume their resources till their limit is reached (in case of storage they can also reclaim these resources back anonymously). We present a definition of such *anonymous yet authorized and bounded cloud resource schemes* along with an instantiation based on Camenisch-Lysyanskaya signatures. Then, we extend the scheme to another scheme providing even more privacy for users (by even hiding the issued resource bound during interactions and thus providing full anonymity to users) and provide some useful extensions for both schemes. We also underpin the practical efficiency of our schemes by means of experimental results obtained from an implementation.

Keywords. Anonymous resource consumption, privacy, cloud computing, Camenisch-Lysyanskaya signatures, proofs of knowledge.

1 Introduction

Cloud computing is an emerging paradigm, but some significant attention remains justifiably focused on addressing security and privacy concerns. Reasons are among others that customers have to trust the security mechanisms and configuration of the cloud provider and the cloud provider itself. Recently, different cryptographic solutions to improve privacy have been proposed, which mainly focus on private storage, private computations and private service usage.

Storing data encrypted seems to be sine qua non in many cloud storage settings, since cloud providers, having access to the storage infrastructure, can neither be considered as fully trustworthy nor are resistant to attacks. Kamara and Lauter [26] propose several architectures for cryptographic cloud storage and provide a sound overview of recent non-standard cryptographic primitives like searchable encryption and attribute-based encryption, which are valuable tools in this context. Other issues are data privacy and verifiability when outsourcing data and performing computations on these data using the cloud as computation infrastructure. The recent introduction of fully homomorphic encryption [24] is a promising concept for performing arbitrary computation on encrypted data. However, up to now these concepts are far from being practical [25]. Another interesting issue from a privacy perspective is to hide user's usage behavior (access patterns and frequencies) when accessing services, e.g. cloud storage services. More precisely, users may not want the cloud provider to learn how often they use a service or which resources they access, i.e. they want to have anonymous and unlinkable access. Nevertheless, cloud providers can be assumed to have access restricted to authorized users and additionally users may want to enforce (attribute-based) access control policies. There are quite different existing approaches to realize this, e.g. one can employ anonymous credential systems [4] or oblivious transfer [8, 9].

In this paper we introduce an additional aspect which may be valuable when moving towards privacy friendly cloud computing and seems to be valuable when used in conjunction with the aforementioned approaches. In particular, we focus on the anonymous yet authorized and bounded use of cloud resources like CPU time (e.g. CPU per hour) or storage space. Although we illustrate our concept by means of the resource storage space in this paper, one may use this approach for arbitrary resources.

Our contribution. We consider a setting where users should be able to register and obtain a resource bound (limit) from a cloud provider (CP) in form of a partially blindly signed token. This token includes an identifier, the already consumed resources and the limit, wheres the limit in fact is the only value signed in clear. This limit determines how much of a resource, e.g. CPU time, storage space, a user is allowed to consume. Then, users should be able to consume their resources in an anonymous and unlinkable yet authorized fashion. For instance, if a user wants to consume l resources, he has to convince the CP that he possesses a signed token with a valid identifier (double-spending protection) and that his consumed resources (including l) do not exceed his bound. If this holds, the anonymous user is allowed to consume the resources and obtains an updated signature for a token corresponding to a new identifier and updated consumed resources. Note, due to the anonymity and unlinkability properties, the CP is unable to track how much a user has already consumed, however, can be sure that he solely consumes what he has been granted. Furthermore, when the resource represents storage space a user may also reclaim resources when he deletes data, whereas these actions should also be anonymous and unlinkable.

We for the first time consider this problem and provide a definition for the concept of anonymous yet authorized and bounded cloud resource schemes. Furthermore, we present an efficient instantiation of such schemes. Then, we extend the scheme to another scheme providing even more privacy for users (by even hiding the issued resource bound during interactions) and provide some useful extensions for both schemes. Our schemes are obtained using recent signature schemes due to Camenisch and Lysyanskaya [12, 13] along with efficient zero-knowledge proofs for proving properties of signed messages. We note that many of the approaches discussed subsequently employ similar features of CL signatures as our approach does. But the signer controlled interactive update of signed messages discussed in Section 4.1, which is an important functionality behind our protocols, seems to be novel. Furthermore, we note that we base our concrete scheme on groups of known order and the essential ingredient is the pairing based CL signature scheme [13]. We want to emphasize that one could as well base the construction on hidden order groups. Then, one can use the strong RSA based CL signature [12], Fujisaki-Okamoto commitments [23] and the range proof proposed by Boudot [7]. But within our construction we achieve much shorter proofs and signatures.

Related work. Pairing based CL signatures [13] and it's strong RSA based pendant [12] are useful to construct various privacy enhancing cryptographic protocols. Among them are anonymous credential systems and group signatures [13] as well as privacy protecting multi-coupon systems [16, 18], anonymous subscriptions [6], electronic toll pricing [5], e-cash systems [11] and n-times anonymous authentication schemes [10] based on compact e-cash or unclonable group identification schemes [21] which achieve similar goals as in [10]. To solve our problem, the most straightforward solutions seems e-cash, i.e. CP issues k coins to a user and a user can use one coin per resource unit. However, to achieve a suitable granularity this induces a large amount of "small valued coins" which makes this approach impractical. The same holds for compact e-cash schemes [11], where a user can withdraw a wallet of 2^l coins at a time and thus the withdrawal procedure is much more efficient. However, in compact e-cash coins from the wallet can only be spend one by one and the above problem still exists. In divisible e-cash [15, 3], which allows a user to withdraw a wallet of value 2^l in a single withdraw protocol, spending a value 2^m for $m \leq l$ can be realized more efficient than repeating the spending 2^m times. However, in the former solution even for a moderate value of $l = 10$ the spending of a single coin requires 800 exponentiations which makes it very expensive. The latter approach is more efficient, but statistical, meaning that users can spend more money than what they withdraw.

Multi-coupons [16, 18] represent a collection of coupons (or coins or tokens) which is issued in a single withdraw protocol and every single coupon of the MC can be spend in an anonymous and unlinkable fashion. But in our scenario, they suffer from the same problem as e-cash.

Recently, Camenisch et al. proposed an interesting protocol for unlinkable priced oblivious transfer with rechargeable wallets [9]. This does not exactly fit our scenario but could be mapped to it. However, [9] do not provide an efficiency analysis in their work and their protocols seem to be quite costly. Their rechargeable wallets are an interesting feature and such an idea is also supported by our second scheme.

2 Definition

Below, we present a description of the problem along with a model of anonymous yet authorized and bounded cloud resource scheme. We note, that we do not provide formal security arguments for the presented schemes in this extended abstracts, but they are presented in the full version.

2.1 Problem Description and Motivation

In our setting we have a cloud provider (CP) and a set of users U. Our main goal is that users are able to purchase a contingent of resources (we focus on storage space here) and CP does not learn anything about the resource consumption behavior of users. In particular, users can store data at the CP as long as there are still resources from their contingent available. The CP is in any interaction with the user convinced that a user is allowed to consume (or reclaim) resources and cannot identify the user nor link any of the user's actions. Clearly, if the resource is storage space and the data objects contain information on the user then this may break the anonymity property. Nevertheless, then we can assume that data is encrypted which seems to be sine qua non in many cloud storage settings.

Our main motivation is that it is very likely that only a few large cloud providers will own large portions of the infrastructure of the future Internet. Thus, these cloud providers will eventually be able to link data and information about resource consumption behavior of their consumers (users) allowing them to build extensive dossiers. Since for many enterprises such a transparency can be too intrusive or problematic if these information are available to their competitors we want to hide these information from cloud providers. As for instance argued in [19], activity patterns may constitute confidential business information and if divulged could lead to reverse-engineering of customer base, revenue size, and the like.

2.2 Definition of the Scheme

An anonymous yet authorized and bounded cloud resource scheme is a tuple (`ProviderSetup`, `ObtainLimit`, `Consume`, `Reclaim`) of polynomial time algorithms or protocols between users U and cloud provider CP respectively:

- `ProviderSetup`. On input a security parameter k, this algorithms outputs a key pair sk and pk of a suitable signature scheme and an empty blacklist BL (for double-spending detection).
- `ObtainLimit`. In this protocol a user u wants to obtain a token t for a resource limit of L units from the CP. The user's output is a token t with corresponding signature σ_t issued by CP. The token contains the limit L and the actually consumed resources s (wheres both may be represented by a single value $L' := L - s$). The output of CP is a transcript T_{OL} of the protocol.
- `Consume`. In this protocol user u wants to consume l units from his remaining resources. The user shows value $t.id$ of a token t and convinces the CP that he holds a valid signature σ_t for token t. If the token was not already spend ($t.id$ is not contained in BL), the signature is valid and there are still enough resources left, i.e. $s' + l \leq L$ (or $L' - l \geq 0$), then the user's output is `accept` and an updated token t' for resource limit L and actually consumed resources $s' + l$ (or $L' - l$) with an updated signature $\sigma_{t'}$ from CP. Otherwise the user's output is `reject`. The output of CP is a transcript T_C.
- `Reclaim`. In this protocol user u wants to reclaim l units, e.g. he wants to delete some data of size l. The protocol is exactly the same as the `Consume` protocol. Except for the `accept` case the updated token t' contains $s' - l$ (or $L' + l$) as the actually consumed resources and the transcript is denoted as T_R. We emphasize that u needs to prove by some means that he is allowed to reclaim l resources, e.g. when deleting some data, the user needs prove knowledge of some secret associated with the data during the integration. Otherwise, users could simply run arbitrary many `Reclaim` protocols to illicitly reclaim resources and indirectly improve their actual resource limit.

3 Preliminaries

An essential ingredient for our construction are honest-verifier zero-knowledge proofs of knowledge (Σ-protocols). We use the notation from [14], i.e. a proof of knowledge of a discrete logarithm $x = \log_g y$ to the base g will be denoted as $PK\{(\alpha) : y = g^\alpha\}$, whereas Greek letters always denote values whose knowledge will be proven. We note, that compositions of single Σ-protocols using conjunctions and disjunctions can be efficiently realized [20]. Furthermore, the non-interactive version of a (composed) proof obtained by applying the Fiat-Shamir transform [22] is denoted as a signature of knowledge or SPK for short.

3.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_t$ be two groups of prime order p, let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_t$ a bilinear map between these two groups. The map e must satisfy the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: There is an efficient algorithm to compute $e(u, v)$ for any $u, v \in \mathbb{G}$.

Though the group operation in $\mathbb{G}$ is in general an additive one, we express both groups using multiplicative notation. This notion is commonly used, since $\mathbb{G}_t$ is always multiplicative and it is more easy to capture the sense of cryptographic protocols.

3.2 Pedersen Commitments

Pedersen commitments [30] represent a widely used commitment scheme working in any group $\mathbb{G}$ of prime order p. Let g, h be random generators of $\mathbb{G}$, whereas $\log_g h$ is unknown. To commit to a value $s \in \mathbb{Z}_p$, one chooses $r \in_R \mathbb{Z}_p$ and computes $C(s, r) = g^s h^r$, which unconditionally hides s as long as r is unknown. To open the commitment, one simply publishes $(s, r, C(s, r))$ and one verifies whether $g^s h^r = C(s, r)$ holds. For simplicity, we often write $C(s)$ for a commitment to s instead of $C(s, r)$. We note that the Pedersen commitment inherits an additive homomorphic property, i.e. given two commitments $C(s_1, r_1) = g^{s_1} h^{r_1}$ and $C(s_2, r_2) = g^{s_2} h^{r_2}$ then one is able to compute $C(s_1 + s_2, r_1 + r_2) = C(s_1, r_1) \cdot C(s_2, r_2)$ without either knowing any of the hidden values s_1 or s_2. Furthermore, note that a proof of knowledge $PK\{(\alpha, \beta) : C = g^\alpha h^\beta\}$ of the ability to open a Pedersen commitment can be realized using a proof of knowledge of a DL representation of C with respect to the elements g and h [28].

3.3 Range Proofs

An elegant proof that a number hidden within a Pedersen commitment lies in an interval $[a, b]$ in the setting of prime order groups was presented in [27]. Although this proof might be impractical in general, since it requires $O(\log\ b)$ single bit-proofs, it is efficient for the application that we have in mind due to small values of b. The basic idea is to consider for a number $x \in [0, b]$ its binary representation $x = x_0 2^0 + x_1 2^1 + \ldots + x_{k-1} 2^{k-1}$, whereas $x_i \in \{0, 1\}$, $0 \leq i < k$. Thereby, $k = [\log_2\ b] + 1$ represents the number of digits, which are necessary to represent every number within $[0, b]$. Now, in essence one proves that the binary representation of x lies within the interval $[0, 2^k - 1]$. This can be done by committing to each x_i using an Okamoto commitment [29] (essentially a Pedersen bit commitment) along with a proof that this commitment hides either 0 or 1 and demonstrating that for commitments to x and all x_i's it holds that $x = x_0 2^0 + x_1 2^1 + \ldots + x_{k-1} 2^{k-1}$. The concrete range proof is a Σ-protocol for a proof of knowledge

$$PK\{(\alpha_0, \ldots, \alpha_{k-1}) : \bigwedge_{i=0}^{k-1} (C_i = h^{\alpha_i} \vee C_i g^{-1} = h^{\alpha_i})\}$$

or $PK\{(\alpha, \beta) : C = g^\alpha h^\beta \wedge\ (0 \leq \alpha \leq b)\}$ for short.

3.4 Camenisch-Lysyanskaya Signature Scheme

Camenisch and Lysyanskaya have proposed a signature scheme in [13] which satisfies the usual correctness and unforgeability properties of digital signatures and is provably secure under the LRSW assumption for groups with bilinear maps, which implies that the DLP is hard (cf. [13]). We present the CL signature scheme below:
Key Generation. Let $\mathbb{G}$ and $\mathbb{G}_t$ be groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_t$ a bilinear map. Choose $x, y, z_1, \ldots, z_l \in_R \mathbb{Z}_p$. The private key is $sk = (x, y, \{z_i\})$ and the public key is $pk = (X, Y, \{Z_i\}, e, g, \mathbb{G}, \mathbb{G}_t, p)$, whereas $X = g^x$, $Y = g^y$ and $Z_i = g^{z_i}$.
Signing. On input message $(m_0, \ldots, m_l)$, sk and pk, choose $a \in_R \mathbb{G}$, compute $A_i = a^{z_i}$, $b = a^y$, $B_i = (A_i)^y$ and $c = a^{x+xym_0} \prod_{i=1}^{l} A_i^{xym_i}$. Output the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$.
Verification. On input of $(m_0, \ldots, m_l)$, pk and $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ check whether

- A_i's are formed correct: $e(a, Z_i) = e(g, A_i)$
- b and B_i's are formed correct: $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$
- c is formed correct: $e(X, a) \cdot e(X, b)^{m_0} \prod_{i=1}^{l} e(X, B_i)^{m_i} = e(g, c)$

What makes this signature scheme particularly attractive is that it allows a receiver to obtain a signature on committed messages (using Pedersen commitments), while the messages are information-theoretically hidden from the signer (messages here means elements of the message tuple). Additionally, the receiver can randomize a CL signature such that the resulting signature is unlinkable to the original signature. Furthermore, receivers can use efficient zero-knowledge proofs to prove knowledge of a signature on committed messages. We will elaborate on the aforementioned functionalities more detailed in Section 4.1 and will show how to extend this functionality to interactive updates of signatures, the signed commitments and messages respectively.

4 Scheme

In this section we present our scheme along with an optional modification in order to increase the privacy in some settings even further. We start with the presentation of an important observation of CL signatures which is central to our constructions. Then, we first give a high level description followed by a detailed description of the schemes. Additionally, we present an performance evaluation of a prototypical implementation which supports the efficiency of the schemes. Finally, we present some extensions as well as system issues.

4.1 Interactive Update of Signed Messages

As already noted, CL signatures allow signing of committed messages (using Pedersen commitments), while the signer does not learn anything about them. Assume that the signer holds a private key $sk = (x, y, z)$ and publishes the corresponding public key $pk = (X, Y, Z, e, g, \mathbb{G}, \mathbb{G}_t, p)$.
Blind signing. If a receiver wants to obtain a blind signature for message m, he chooses $r \in_R \mathbb{Z}_p$, computes a commitment $C = g^m Z^r$ and sends C along with a signature of knowledge $SPK\{(\alpha, \beta) : C = g^\alpha Z^\beta\}$ to the signer (the ability to open the commitment is necessary for the security of the scheme, cf. [13]). If the verification of the proof holds, the signer computes a signature $\sigma = (a, A, b, B, c)$ for the commitment C by choosing $k \in_R \mathbb{Z}_p$, setting $a = g^k$ and computing $\sigma = (a, a^z, a^y, a^{yz}, a^x C^{kxy})$ and sends σ to the receiver.
Verification. In order to show the signature to a verifier, the receiver randomizes the signature by choosing $\mathsf{r}, \mathsf{r}' \in_R \mathbb{Z}_p$ and computing $\sigma' = (a', A', b', B', c')$ as $\sigma' = (a^{\mathsf{r}}, A^{\mathsf{r}}, b^{\mathsf{r}}, B^{\mathsf{r}}, c^{\mathsf{r}\mathsf{r}'})$ and sends σ' with the message m along with a signature of knowledge $SPK\{(\gamma, \delta) : v_\sigma^\gamma = \mathbf{v} v_r^\delta\}$ to the verifier. Therefore, both need to compute $v_\sigma = e(c', g)$, $\mathbf{v} = e(X, a') \cdot e(X, b')^m$ and $v_r = e(X, B')$. The verifier checks the proof and checks whether A' as well as b' and B' were correctly formed. Note, that the proof can be conducted by means of a standard DL-representation proof [17], which can easily be seen by rewriting the proof as $SPK\{(\gamma, \delta) : \mathbf{v} = v_\sigma^\gamma (v_r^{-1})^\delta\}$.
Remark. Observe, that we can realize a concept which is similar to partially blind signatures. However, in contrast to existing partially blind signature schemes [1], where the signer can integrate some common agreed upon information in the signature, here, the signer *arithmetically*

adds a message to the "blinded message" (hidden in the commitment). Therefore, during the signing, the signer simply updates the commitment to $C' = Cg^{m_S}$ and uses C' instead of C during signing. The receiver then obtains a signature for message $m + m_S$, whereas m_S is determined by the signer and m is hidden from the signer.

Update. The interesting and from our point of view novel part is that a signer can use a somewhat related idea to "update" a randomized signature without showing the message. Assume that a receiver holds a randomized signature σ' for message (m', r) whereas $m' = m + m_S$ and wants the signer to update the signature such that it represents a signature for message $(m' + m'_S, r + 1)$. Since showing m', as within the signature above, would destroy the unlinkability due to both messages are known, the receiver can solely prove that he knows the message in zero knowledge and both can then interactively update the signature. Therefore in the verification the receiver provides a signature of knowledge $SPK\{(\alpha, \beta, \gamma) : v_\sigma^\alpha = \mathbf{v} v_{m'}^\beta v_r^\gamma\}$ to the verifier, whereas $v_\sigma = e(g, c')$, $\mathbf{v} = e(g, a')$, $v_{m'} = e(g, b')$ and $v_r = e(g, B')$, which convinces the signer that the receiver possesses a valid signature for unknown message (m', r). Then, for the update, i.e. to *add* m'_S it is sufficient for the signer to compute $\tilde{C}_{m'+m'_S} = a'^{m'_S} A'$ and send it to the receiver. The receiver computes $C_{m'+m'_S} = (\tilde{C}_{m'+m'_S})^{\mathsf{r}'}$ and provides a signature of knowledge $SPK\{(\alpha, \beta, \gamma) : v_\sigma^\alpha = \mathbf{v} v_{m'}^\beta v_r^\gamma \ \wedge \tilde{C}_{m'+m'_S} = (C_{m'+m'_S})^\alpha\}$. Note that this proof convinces the signer that the receiver has randomized the commitment of the signer using the same random factor (r') as within the randomization of the signature. Then, the signer computes the updated signature $\sigma'' = (a^{\tilde{\mathsf{r}}}, A''^{\tilde{\mathsf{r}}}, b'^{\tilde{\mathsf{r}}}, B''^{\tilde{\mathsf{r}}}, (c'(C_{m'+m'_S})^{xy})^{\tilde{\mathsf{r}}})$ for $\tilde{\mathsf{r}} \in \mathbb{Z}_p$ and gives $\sigma'' = (a'', A'', b'', B'', \tilde{c}'')$ to the receiver. The receiver sets $c'' = (\tilde{c}'')^{\mathsf{r}'^{-1}}$ and now holds a valid signature for message $(m' + m'_S, r + 1)$ which he can in turn randomize. Therefore, observer that in the signature tuple only the last element actually includes the messages and we have $c' = c^{\mathsf{r}\mathsf{r}'} = (a^x C'^{kxy})^{\mathsf{r}\mathsf{r}'} = (a^{x+xy(m'+zr)})^{\mathsf{r}\mathsf{r}'}$ and $(C_{m'+m'_S})^{xy} = (a^{xy(m'_S+z)})^{\mathsf{r}}$. By taking these results together we have a well formed signature component $c'' = (a^{x+xy(m'+m'_S+z(r+1))})^{\mathsf{r}\mathsf{r}'}$. The remaining elements of the signature are easy to verify for correctness.

Remark. This functionality can easily be extended to signatures on arbitrary tuples of messages, will be a building block for our scheme and may also be of independent interest. Note that issuing a new signature in every step without revealing the hidden messages would not work and thus we use this "update functionality".

4.2 High Level Description of the First Scheme

Before presenting the detailed protocols, we provide a high level description. The aim of our construction is to let the user solely prove in each `Consume` protocol that enough storage space is available. In this setting, the user does not provide any useful information about the actual consumed space to the verifier, but the verifier learns only the fact that the user is still allowed to consume storage space.

ProviderSetup. The cloud provider generates a key-pair (sk, pk) for the CL signature scheme, publishes pk, initializes an empty blacklist BL and fixes a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of space limits.

ObtainLimit. A user chooses a limit $L \in \mathcal{L}$ and obtains a CL signature σ_t for a token $t = (C(id), C(s), L)$, whereas the initially consumed storage space is set to be $s = 1$.

Consume. Assume that the user holds a token signature pair $t = ((C(id), C(s), L), \sigma_t)$. Note, that id (the token-id) and s were signed as commitments and thus the signer is not aware of these values. When a user wants to integrate a data object d, the user computes $C(id')$ for the new token, randomizes the signature σ_t to σ'_t and proves that σ'_t is a valid signature for id and L (by revealing these two elements) and an unknown value s that satisfies $(s + |d|) \in [0, L]$ or equivalently $s \in [0, L - |d|]$, i.e. when integrating the new data object d the user needs to prove that after adding of $|d|$ space units at most L storage space will be consumed. If id is not contained in BL and this proof succeeds, the signature will be updated to a signature for $C(id + id')$, $C(s + |d|)$ and L. Consequently, the provider adds id to BL and the user obtains an updated signature for a token $t' = (C(id + id'), C(s + |d|), L)$. Otherwise, the cloud provider

will reject the integration of a new data object.

Reclaim. Assume that the user holds a token signature pair $t = ((C(id), C(s), L), \sigma_t)$. When a user wants to delete a data object d, as above, the user computes $C(id')$ for the new token, randomizes the signature σ_t to σ_t' and proves that he is allowed to delete d and that σ_t' is a valid signature for id and L (by revealing these two elements). If id is not contained in BL and the signature is valid, the user obtains a signature for a token $t' = (C(id + id'), C(s - |d|), L)$. Otherwise, the cloud provider will reject to delete d.

4.3 Detailed Description of the First Scheme

Subsequently, we provide a more detailed description of our protocols providing the technical details.

ProviderSetup: The cloud provider generates a key-pair for the CL signature scheme to sign tokens of the form $t = (id, s, L)$. More precisely, the cloud provider signs tokens of the form $t = (id, r_{id}, s, r_s, L)$, but we usually omit the randomizers for the ease of presentation. Consequently, the cloud provider obtains the private key $sk = (x, y, z_1, z_2, z_3, z_4)$ and publishes the public key $pk = (X, Y, Z_1, Z_2, Z_3, Z_4, e, g, \mathbb{G}, \mathbb{G}_t, p)$. Furthermore, he initializes an empty blacklist BL and fixes a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of limits that can be obtained by users.

ObtainLimit: A user registers with the cloud provider and obtains a space limit $L_i \in \mathcal{L}$ (we do not fix any concrete protocol for this task here since no anonymity is required). After the user has registered and both have agreed on the value L_i (which we denote as L below for simplicity), they proceed as depicted in Protocol 1.

1. The user chooses a token-identifier $id \in_R \{0,1\}^{l_{id}}$ and randomizers $r_{id}, r_s \in_R \mathbb{Z}_p$ for the commitments and we let the user start with value $s = 1$. Then, he computes the commitments $C_{id} = g^{id} Z_1^{r_{id}}$ and $C_s = Z_2^s Z_3^{r_s}$ and sends them along with a signature of knowledge

$$SPK\{(\alpha, \beta, \gamma) : \; C_{id} = g^{\alpha} Z_1^{\beta} \wedge C_s = Z_2 Z_3^{\gamma}\} \tag{1}$$

   to prove the ability to open the commitments, whereas the second part in the proof also convinces the cloud provider that $s = 1$.
2. If the verification of the signature of knowledge in (1) holds, the cloud provider computes a CL signature for (C_{id}, C_s, L) as follows: He chooses $k \in_R \mathbb{Z}_p$, computes $a = g^k$, $b = a^y$, $A_i = a^{z_i}$, $B_i = A_i^y$ for $1 \leq i \leq 4$ and $c = a^x (C_{id} C_s Z_4^L)^{kxy}$ and sends $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ to the user.
3. The user verifies whether the signature is valid and if this holds the user is in possession of a valid signature σ for a token $t = (id, s, L)$, whereas the cloud provider is not aware of id and knows that $s = 1$. Furthermore, the user locally randomizes the signature σ to $\sigma' = (a', \{A_i'\}, b', \{B_i'\}, c')$ by choosing $\mathsf{r}, \mathsf{r}' \in \mathbb{Z}_p$ and computing $\sigma' = (a^{\mathsf{r}}, \{A_i^{\mathsf{r}}\}, b^{\mathsf{r}}, \{B_i^{\mathsf{r}}\}, c^{\mathsf{r}\mathsf{r}'})$.

Remark. All further actions are fully anonymous and in practice also unlinkable, since we can assume that one limit will be issued to a quite large number of users (and the limit is the only information that could potentially be used for linking)!

Prot. 1: The `ObtainLimit` protocol.

Consume: A user holds a randomized signature $\sigma' = (a', \{A_i'\}, b', \{B_i'\}, c')$ for a token $t = (id, s, L)$ and wants to integrate a data object d. The protocol to integrate a data object and obtain a new token is depicted in Protocol 2.

1. The user sends the randomized signature σ', the "visible part" (id, L) of the token t and a data object d along with a signature of knowledge

$$SPK\{(\alpha, \beta, \gamma, \delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta \ \wedge \ (0 \leq \gamma \leq 2^{l_L - l_{|d|}} - 1)\} \tag{2}$$

for the validity of the randomized signature containing a proof that still enough space is available to the cloud provider. It must be noted, that the presentation of the proof in (2) represents a shorthand notation for the signature of knowledge

$$\begin{aligned} SPK\{(\alpha, \beta, \gamma, \delta, \epsilon, \epsilon_1, \ldots, \epsilon_k, \zeta, \zeta_1, \ldots, \zeta_k) : \ \mathbf{v} = v_\sigma^\alpha (v_{r_{id}}^{-1})^\beta (v_s^{-1})^\gamma (v_{r_s}^{-1})^\delta \ \wedge \\ C = g^\beta Z_1'^\zeta \ \wedge \\ C = \prod_{i=1}^{k} (g^{\epsilon_i} Z_1'^{\zeta_i})^{2^{i-1}} \ \wedge \\ \bigwedge_{i=1}^{k} (C_i = Z_1^{\zeta_i} \vee C_i g^{-1} = Z_1'^{\zeta_i})\} \end{aligned}$$

Essentially, besides the DL-representation proof for the validity of the randomized signature, we use an additional commitment $C = g^s Z_1'^r$ to the value s with a new randomizer r computed as

$$r = r_1 2^0 + r_2 2^1 + \ldots + r_k 2^{k-1} \ \text{MOD} \ p$$

for r_i's chosen uniformly at random from $\mathbb{Z}_p$ and the single commitments for the range proof are $C_i = g^{s_i} Z_1'^{r_i}$. It also must be mentioned, that k represents $l_L - l_{|d|}$, the binary length of $L - |d|$. Furthermore, note that in case of $s = 1$, i.e. in the first execution of the `Consume` protocol, it would not be necessary to provide a range proof. However, when performing a range proof, the initial `Consume` protocol is indistinguishable from other protocol executions and thus provides stronger privacy guarantees.
2. The cloud provider checks whether $id \notin BL$. If id is not blacklisted, the cloud provider verifies the validity of the signature for the part (id, L) of the token t. Therefore, the cloud provider locally computes the values

$$v_\sigma = e(g, c'),\ v_{r_{id}} = e(X, B_1'),\ v_s = e(X, B_2'),\ v_{r_s} = e(X, B_3') \text{ and}$$

$$\mathbf{v} = e(X, a') \cdot e(X, b')^{id} \cdot e(X, B_4')^L$$

from pk, (id, L) and σ' and verifies the signature of knowledge (2) Additionally, he checks whether the A_i''s as well as b' and B_i''s are correctly formed.
3. A positive verification convinces the cloud provider that enough storage space is available to integrate d and a signature for an updated token t' can be computed in cooperation with the user as follows: Firstly, we need an observation regarding the signature σ'. Note, that the only element of the signature that depends on the message is c', which can be rewritten as

$$c' = (a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)})^{\mathsf{r}'} = (a^{x+xyid} A_1^{xyr_{id}} A_2^{xys} A_3^{xyr_s} A_4^{xyL})^{\mathsf{r}'}$$

and in order to update a signature for the id-part (to construct a new id for the new token) it is sufficient to update a and A_1. To update the s-part, which amounts to update the currently consumed space, it is sufficient to update A_2 and A_3. The latter update needs to be computed by the cloud provider to be sure that the correct value $|d|$ is integrated and the former one needs to be computed by the user to prevent the cloud provider from learning the new token identifier. Hence, the cloud provider computes $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$ and sends $\tilde{C}_{s+|d|}$ to the user, who verifies whether $|d|$ has been used to update the commitment. The user in turn chooses a new identifier and randomizer $id', r_{id'} \in_R \mathbb{Z}_p$, computes $C_{id+id'} = (a'^{id'} A_1'^{r_{id'}})^{\mathsf{r}'}$, $C_{s+|d|} = (\tilde{C}_{s+|d|})^v = (A_2'^{|d|} A_3')^{\mathsf{r}'}$ and sends $(C_{id+id'}, C_{s+|d|})$ along with a signature of knowledge:

$$\begin{aligned} SPK\{(\epsilon, \zeta, \eta, \phi, \iota, \kappa) : C_{id+id'} = a'^\epsilon A_1'^\zeta \ \wedge \\ \tilde{C}_{s+|d|} = (C_{s+|d|})^\eta \ \wedge \ \mathbf{v} = v_\sigma^\eta (v_{r_{id}}^{-1})^\phi (v_s^{-1})^\iota (v_{r_s}^{-1})^\kappa\} \end{aligned}$$

to the cloud provider. Note, that the user additionally to the knowledge of the ability to open the commitments proves that he has randomized the commitment $\tilde{C}_{s+|d|}$ to a commitment $C_{s+|d|}$ using the same randomization factor (r') as used to randomize the signature σ without revealing this value. After positive verification of this signature of knowledge, the cloud provider chooses $\tilde{\mathsf{r}} \in_R \mathbb{Z}_p$ and computes an updated signature

$$\sigma'' = (a'^{\tilde{\mathsf{r}}}, \{A_i'^{\tilde{\mathsf{r}}}\}, b'^{\tilde{\mathsf{r}}}, \{B_i'^{\tilde{\mathsf{r}}}\}, (c'(C_{id+id'} C_{s+|d|})^{xy})^{\tilde{\mathsf{r}}}) \tag{3}$$

and sends this updated signature $\sigma'' = (a'', \{A_i''\}, b'', \{B_i''\}, \tilde{c}'')$ to the user. The user sets $c'' = (\tilde{c}'')^{\mathsf{r}'^{-1}}$ and obtains a valid signature for a token $t' = (id + id', s + |d|, L)$ or more precisely a token $t' = (id + id', r_{id} + r_{id'}, s + |d|, r_s + 1, L)$, which he verifies for correctness (in Appendix A we show that σ'' is indeed a valid signature). Consequently, the user can randomize σ'' and run a new `Consume` protocol for a data object d' with token $t' = (id + id', s + |d|, L)$.

Prot. 2: The `Consume` protocol.

Reclaim: Reclaiming resources, i.e. deleting a data object, is achieved by a slight adaption of the `Consume` protocol. In step 1, instead of the SPK (2) the user provides the subsequent

signature of knowledge (the proofs that enough space is available is not necessary)

$$SPK\{(\alpha,\beta,\gamma,\delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta\}$$

And in step 3, the cloud provider computes $\tilde{C}_{s-|d|} = A_2'^{p-|d|} A_3'$ instead of $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$.

Remark. As we have already mentioned, a cloud provider should only perform a `Reclaim` protocol if the user is able to prove the possession of the data object d (and we may assume that only owners delete their data objects). It is not the focus of this paper to provide a solution to this task. However, a quite straightforward solution would be to commit to some secret value for every data object and the cloud provider requires a user to open the commitment or prove knowledge that he is able to open the commitment to delete a data object. This problem is somewhat similar to what Halevi et al. recently denoted as proofs of ownership (PoWs) [32]. However, their focus although also in the cloud storage setting is (client-side) deduplication, i.e. storing exactly the same file only once thus avoiding unnecessary copies of repeating data. This is usually achieved by sending a hash of a file and the cloud checks if this hash is already registered. In order to avoid security problems in this setting, they employ PoWs which require users to prove that a user actually holds a file. This is somewhat similar but diametric to proofs of data possession (PDPs) [2] and proofs of retrievability (PORs) [31] respectively.

4.4 A Modified Scheme (Scheme 2) Providing Even more Privacy for Users

In order to increase privacy further, it may be desirable that the initially issued limit L is hidden from the CP during `Consume` or `Reclaim` protocols. We, however, note that if the number of initial tokens associated to CP-defined limits in $\mathcal{L}$ is huge, the respective anonymity sets may be of reasonable size for practical application and this adaption may not be necessary. Nevertheless, we provide an adaption of our protocols which removes the necessity to include L, does only include the available amount of resources (denoted as s) and hides this value s from the CP during any further interactions. We present the modification below:

ProviderSetup. Now, tokens are of the form $t = (id, r_{id}, s, r_s)$ and thus the private key is $sk = (x, y, z_1, z_2, z_3)$ and the public key is $pk = (X, Y, Z_1, Z_2, Z_3, e, g, \mathbb{G}, \mathbb{G}_t, p)$.

ObtainLimit. The user computes commitments $C_{id} = g^{id} Z_1^{r_{id}}$ and $C_s = Z_3^{r_s}$ and provides $SPK\{(\alpha,\beta,\gamma) : C_{id} = g^\alpha Z_1^\beta \wedge C_s = Z_3^\gamma\}$. The element c of the signature is now computed by the CP as $c = a^x (C_{id} C_s Z_2^L)^{kxy}$ and the user can randomize this signature for token $t = (id, r_{id}, L, r_s)$ as usual.

Consume. Here the user only provides id of the actual token and a signature of knowledge

$$SPK\{(\alpha,\beta,\gamma,\delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta \wedge \ (2^{l_{|d|}} - 1 \leq \gamma \leq 2^{l_\mathbf{L}} - 1)\}$$

In this setting $\mathbf{L}$ does not represent a user-specific limit but *the maximum* of all issued limits, whereas this proof convinces the CP that enough resources to integrate d are still available (note that the local computations of the CP for the verification of the signature in step 2 have to be adapted, which is however straightforward). In step 3, the update of the signature remains identical to the first scheme with the exception that the CP computes the commitment as $\tilde{C}_{s-|d|} = A_2'^{p-|d|} A_3'$, which updates the remaining resources, e.g. in the first run of the `Consume` protocol to $s := L - |d|$.

Reclaim. The reclaim protocol remains identical to the first scheme with the exception that $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$.

4.5 Performance Evaluation

In this section we provide a performance evaluation of our first scheme. We have implemented the user's and the cloud provider's parts of the protocols in Java using the `jPBC`[1] library version 1.2.0 written by Angelo De Caro. This library provides a Java porting of as well as a Java wrapper

[1] `http://libeccio.dia.unisa.it/projects/jpbc/`

for the Pairing-Based Cryptography Library (PBC)[2] developed by Ben Lynn in C. In particular, we have used the Java PBC wrapper which calls the PBC C library and is significantly faster than the pure Java implementation. All our experiments were performed on an Intel Core 2 duo running at 2.6 GHz with 3GB RAM on Linux Ubuntu 10.10.

As the cryptographic setting we have chosen a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_t$ constructed on the supersingular elliptic curve $y^2 = x^3 + x$ over a prime field $\mathbb{F}_q$ where $|q| = 512$ bits and $q \equiv 3 \pmod 4$. The group $\mathbb{G}$ represents a subgroup of $E(\mathbb{F}_q)$ of order $r = 160$ bits. The embedding degree is $k = 2$ and thus $\mathbb{G}_t$ is a subgroup of $\mathbb{F}_{q^2}$ and with our choice of the parameters we obtain a DL security of 1024 bit. For the non-interactive proofs of knowledge we have used the SHA-256 hash function.

Experiments. Our setting for the experiments is as follows: For the computational performance we have taken the average over 100 experiments, with limits $L = 10^i$, $i = 3, \ldots, 9$ each. Thereby, within every of the 100 experiments per limit, the user has conducted 10 `Consume` as well as 10 `Reclaim` operations with $|d|$ sampled uniformly at random from $[1, 10^{i-2}]$. Figure 1 presents the performance of the `ObtainLimit`, the `Consume` and the `Reclaim` protocols from a computational and bandwidth perspective, whereas point compression for elements in $\mathbb{G}$ is used to reduce the bandwidth consumption. As one can see, all protocols are highly efficient from the user's as

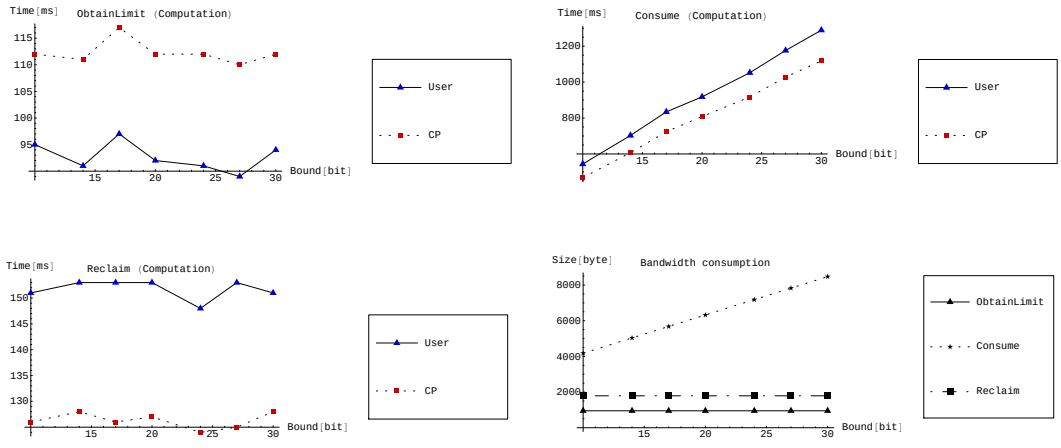

Fig. 1. Experimental results from a Java implementation of our first scheme.

well as the cloud provider's perspective, both in the computational effort and the bandwdith consumption. This holds, although the code has not been optimized for performance and pre-computations have not been used. Hence, our evaluation shows that from the efficiency point of view our protocols are entirely practical.

4.6 Extensions and System Issues

Below, we present two extensions of our schemes which seem to be valuable when deploying them for practical applications.

Limited validity. One could rightly argue that in a large scale cloud the double spending detection of token identifiers using a database (blacklist) does not scale well. In order to overcome this limitation, we can extend our schemes such that a resource limit associated to a token only has a limited validity. Then, before the validity ends a user has to provide the actual token, i.e. the identifier and the available resources (either s and L or solely s in the second scheme) along with the corresponding signature. Then the user runs a new `ObtainLimit` protocol with the CP. Note that in case of the first scheme users should not end up with a new limit L which is very likely to be unique and thus users should take one of the predefined limits. We now sketch how

[2] `http://crypto.stanford.edu/pbc/`

this adaption for the first scheme looks like (for the second one it is analogous): The keys of the CP are adapted such that the public key is $pk = (X, Y, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, e, g, \mathbb{G}, \mathbb{G}_t, p)$. Token are augmented by elements (V, r_V) which represent the validity period, e.g. an encoding in Unix time. In the `ObtainLimit` protocol the user additionally computes $Z_6^{r_V}$ (and proves knowledge of this DL) and the c part of the signature is adapted to $c = a^x(C_{id}C_s Z_4^L Z_5^V Z_6^{r_V})$ whereas the CP here integrates the validity V. The remainig ideas stay the same with exception that in the `Consume` protocol, the SPK needs to be adapted to

$$SPK\{(\alpha, \beta, \gamma, \delta, \epsilon, \zeta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta v_V^\epsilon v_{r_V}^\zeta \wedge$$
$$(0 \leq \gamma \leq 2^{l_L - l_{|d|}} - 1) \wedge (2^{l_{\mathtt{time}}} - 1 \leq \epsilon \leq 2^{l_{\mathbf{p}}} - 1)\}$$

whereas $\mathbf{p}$ represents the maximum validity period and `time` the representation of the actual date and time (in the `Reclaim` we only need the second range proof). For the update of the signature and the token respectively, the user has additionally to compute $C_V = (A'_5 A_6'^{r'_V})^{\mathsf{r}'}$ and augment the prove of knowledge in step 3 of Protocol 2 to

$$SPK\{(\zeta, \eta, \phi, \iota, \kappa, \lambda, \mu, \nu, \xi) : C_{id+id'} = a'^\zeta A_1'^\eta \ \wedge C_V = A_5'^\phi A_6'^\iota \wedge$$
$$\tilde{C}_{s+|d|} = (C_{s+|d|})^\phi \ \wedge \ \mathbf{v} = v_\sigma^\phi (v_{r_{id}}^{-1})^\kappa (v_s^{-1})^\lambda (v_{r_s}^{-1})^\mu (v_V^{-1})^\nu (v_{r_V}^{-1})^\xi\}$$

Note that these modifications do influence the overall performance of the `Consume` protocol approximately by a factor of two, which though performs very good in practice when compared with our experimental results.

Elasticity. Clouds extremely benefit from users being able to request resources "on the fly". In our first scheme this can only be achieved by means of requesting additional tokens, i.e. running additional `ObtainLimt` protocols for the required resource, and users have then to manage a list of tokens. Although this seems to be an issue which can be handled in practical applications, the second scheme (modification to hide the limit L) allows for such updates. Therefore we can simply use the `Reclaim` protocol of Section 4.4 (we may denote it as `Recharge` in this case), whereas $|d|$ is simply replaced by the amount of resources to be extended.

5 Conclusion

In this paper we have investigated the problem of anonymous yet authorized and bounded use of cloud resources, which means that users should be able to register and obtain a resource limit from a cloud provider such that this limit determines how much of a resource, e.g. CPU time, storage space, a user is allowed to consume. Then, users should be able to consume (or reclaim) their resources in an anonymous and unlinkable fashion, but the ability of users to consume resources should be constrained by their issued limit. We have presented a scheme, it's modification providing even more privacy, have presented extensions valuable for practical application and have supported the efficiency of the proposed scheme by a performance analysis based on a prototypical implementation. Concluding we present anonymity revocation as an open problem and then we briefly discuss future work.

Anonymity revocation. It is not clear to us how anonymity revocation could be suitably realized in this setting. We argue that it does not seem to be meaningful to use identity escrow within every transaction, i.e. to employ techniques known from group signatures (which could though be realized using CL signatures). It is absolutely not clear who would have the power to perform anonymity revocation. In contrast, if at all, it seems more suitable to employ techniques like used within e-cash [11] or (n-times) anonymous authentication [11, 21]. Mapped to our scenario this would mean that the identity of an anonymous user is solely revealed if the user tries to consume more resources than allowed (although this is prevented by the protocols). However, it is not clear to us how to achieve this, since in the aforementioned approaches spend protocols or authentications are atomic and in our setting we do not know in advance how often a user will consume or reclaim resources. We leave this functionality as an open problem for future work.

In order to gain more insights into system issues and to gather experience on limitations in practical use we are working on the integration of our schemes with the full functionality (all extensions) into the Eucalyptus cloud[3] using Amazon's S3 storage service to.

References

1. Abe, M., Okamoto, T.: Provably Secure Partially Blind Signatures. In: CRYPTO '00. LNCS, vol. 1880, pp. 271–286. Springer (2000)
2. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X.: Provable Data Possession at Untrusted Stores. In: 14th ACM Conference on Computer and Communications Security, CCS 2007. pp. 598–609. ACM (2007)
3. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Financial Cryptography and Data Security 2008. LNCS, vol. 5143, pp. 287–301. Springer (2008)
4. Backes, M., Camenisch, J., Sommer, D.: Anonymous Yet Accountable Access Control. In: ACM Workshop on Privacy in the Electronic Society, WPES '05. pp. 40–46. ACM (2005)
5. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-Preserving Electronic Toll Pricing. In: 19th USENIX Security Symposium. pp. 63–78. USENIX Association (2010)
6. Blanton, M.: Online Subscriptions with Anonymous Access. In: 3rd ACM Symposium on Information, Computer and Communications Security, ASIACCS '08. pp. 217–227. ACM (2008)
7. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer (2000)
8. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious Transfer with Access Control. In: 16th ACM Conference on Computer and Communications Security, CCS '09. pp. 131–140. ACM (2009)
9. Camenisch, J., Dubovitskaya, M., Neven, G.: Unlinkable Priced Oblivious Transfer with Rechargeable Wallets. In: 14th International Conference on Financial Cryptography and Data Security, FC 2010. LNCS, vol. 6052, pp. 66–81. Springer (2010)
10. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to Win the Clone Wars: Efficient Periodic n-Times Anonymous Authentication. In: 13th ACM Conference on Computer and Communications Security, CCS '06. pp. 201–210. ACM (2006)
11. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: EUROCRYPT '05. LNCS, vol. 3494, pp. 302–321. Springer-Verlag (2005)
12. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Security in Communication Networks, SCN' 02. LNCS, vol. 2576, pp. 268–289. Springer (2002)
13. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer (2004)
14. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). In: CRYPTO '97. LNCS, vol. 1294, pp. 410–424. Springer (1997)
15. Canard, S., Gouget, A.: Divisible E-Cash Systems Can Be Truly Anonymous. In: EUROCRYPT 2007. LNCS, vol. 4515, pp. 482–497. Springer (2007)
16. Canard, S., Gouget, A., Hufschmitt, E.: A Handy Multi-coupon System. In: Applied Cryptography and Network Security 2006. LNCS, vol. 3989, pp. 66–81. Springer (2006)
17. Chaum, D., Evertse, J.H., van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In: EUROCRYPT '87. LNCS, vol. 304, pp. 127–141. Springer (1987)
18. Chen, L., Escalante, A.N., Löhr, H., Manulis, M., Sadeghi, A.R.: A Privacy-Protecting Multi-Coupon Scheme with Stronger Protection Against Splitting. In: Financial Cryptography and Data Security 2007. LNCS, vol. 4886, pp. 29–44. Springer (2007)
19. Chen, Y., Paxson, V., Katz, R.H.: What's New About Cloud Computing Security? Tech. Rep. UCB/EECS-2010-5, University of California, Berkeley (2010)
20. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: CRYPTO '94. LNCS, vol. 839, pp. 174–187. Springer (1994)
21. Damgård, I., Dupont, K., Pedersen, M.Ø.: Unclonable Group Identification. In: EUROCRYPT '06. LNCS, vol. 4004, pp. 555–572. Springer (2006)
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1987)
23. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: CRYPTO '97. LNCS, vol. 1294, pp. 16–30. Springer (1997)

[3] `http://open.eucalyptus.com/`

24. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: 41st Annual ACM Symposium on Theory of Computing, STOC 2009. pp. 169–178 (2009)
25. Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. Cryptology ePrint Archive, Report 2010/520 (2010), http://eprint.iacr.org/
26. Kamara, S., Lauter, K.: Cryptographic Cloud Storage. In: Financial Cryptography Workshops 2010. LNCS, vol. 6054, pp. 136–149. Springer (2010)
27. Mao, W.: Guaranteed Correct Sharing of Integer Factorization with Off-Line Shareholders. In: Public Key Cryptography 1998. LNCS, vol. 1431, pp. 60–71. Springer (1998)
28. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: CRYPTO '92. LNCS, vol. 740, pp. 31–53. Springer (1992)
29. Okamoto, T.: An Efficient Divisible Electronic Cash Scheme. In: CRYPTO '95. LNCS, vol. 963, pp. 438–451. Springer (1995)
30. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: CRYPTO '91. LNCS, vol. 576, pp. 129–140. Springer (1992)
31. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer (2008)
32. Shai Halevi, Danny Harnik, B.P.A.S.P.: Proofs of Ownership in Remote Storage Systems. Cryptology ePrint Archive, Report 2011/207 (2011), http://eprint.iacr.org/

A Correctness of Update and Randomization in Consume

We need to show, that $\sigma'' = (a'', \{A_i''\}, b'', \{B_i''\}, c'')$ is a valid signature for $t' = (id + id', r_{id} + r_{id'}, s + |d|, r_s + 1, L)$. Therefore, we firstly take a closer look at the signature component c'' and then show that the signature verification for σ'' works and consequently σ'' represents a valid CL signature.

Regarding the component c'', observe that we have

$$c = a^x (C_{id} C_s Z_4^L)^{kxy} = a^x (g^{id} Z_1^{r_{id}} Z_2^s Z_3^{r_s} Z_4^L)^{kxy} = a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)}$$

and

$$c' = c^{\mathsf{r}\mathsf{r}'} = (a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)})^{\mathsf{r}\mathsf{r}'}$$

as well as

$$(C_{id+id'} C_{s+|d|})^{xy} = ((a'^{id'} A_1'^{r_{id'}})^{\mathsf{r}'} (A_2'^{|d|} A_3')^{\mathsf{r}'})^{xy} = (a^{xy(id'+z_1 r_{id'}+z_2|d|+z_3)})^{\mathsf{r}\mathsf{r}'}$$

Thus, by taking these results together we obtain

$$\begin{aligned} c'' &= ((c'(C_{id+id'} C_{s+|d|})^{xy})^{\tilde{\mathsf{r}}})^{\mathsf{r}'-1} = (((a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)} a^{xy(id'+z_1 r_{id'}+z_2|d|+z_3)})^{\mathsf{r}\mathsf{r}'})^{\tilde{\mathsf{r}}})^{\mathsf{r}'-1} \\ &= (a^{x+xy(id+id'+z_1(r_{id}+r_{id'})+z_2(s+|d|)+z_3(r_s+1)+z_4 L)})^{\mathsf{r}\tilde{\mathsf{r}}} \end{aligned}$$

and we can write all components of σ'' similar to c''. More precisely, we can represent the remaining components as $(a^{\mathsf{r}\tilde{\mathsf{r}}}, \{A_i^{\mathsf{r}\tilde{\mathsf{r}}}\}, b^{\mathsf{r}\tilde{\mathsf{r}}}, \{B_i^{\mathsf{r}\tilde{\mathsf{r}}}\})$. Now, it remains to show that the verification relations for signature σ'' work. Recall, therefore we need to show that the A_i'''s, b'', B_i'''s and c'' are formed correctly:

- A_i'''s need to satisfy $e(a'', Z_i) = e(g, A_i'')$: This can easily be verified, since

$$e(a'', Z_i) = e(a^{\mathsf{r}\tilde{\mathsf{r}}}, g^{z_i}) = e(g, g^{kz_i \mathsf{r}\tilde{\mathsf{r}}}) = e(g, (a^{z_i})^{\mathsf{r}\tilde{\mathsf{r}}}) = e(g, A_i^{\mathsf{r}\tilde{\mathsf{r}}}) = e(g, A_i'')$$

- b needs to satisfy $e(a'', Y) = e(g, b'')$: This holds, since

$$e(a'', Y) = e(a^{\mathsf{r}\tilde{\mathsf{r}}}, g^y) = e(g^{k\mathsf{r}\tilde{\mathsf{r}}}, g^y) = e(g, g^{ky\mathsf{r}\tilde{\mathsf{r}}}) = e(g, a^{y\mathsf{r}\tilde{\mathsf{r}}}) = e(g, b^{\mathsf{r}\tilde{\mathsf{r}}}) = e(g, b'')$$

- B_i'''s need to satisfy $e(A_i'', Y) = e(g, B_i'')$: This holds, since

$$e(A_i'', Y) = e(A_i^{\mathsf{r}\tilde{\mathsf{r}}}, g^y) = e(g^{kz_i \mathsf{r}\tilde{\mathsf{r}}}, g^y) = e(g, g^{kz_i y\mathsf{r}\tilde{\mathsf{r}}}) = e(g, A_i^{y\mathsf{r}\tilde{\mathsf{r}}}) = e(g, B_i^{\mathsf{r}\tilde{\mathsf{r}}}) = e(g, B_i'')$$

- c'' needs to satisfy

$$e(X, a'') \cdot e(X, b'')^{id+id'} \cdot e(X, B_1'')^{r_{id}+r_{id'}} \cdot e(X, B_2'')^{s+|d|} \cdot e(X, B_3'')^{r_s+1} \cdot e(X, B_4'')^L = e(g, c'')$$

  which holds, since by rewriting the left hand side and using bilinearity we obtain

$$\begin{aligned} &e(g, a)^{\mathsf{r}\tilde{\mathsf{r}}} \cdot e(g, a^{xy(id+id')})^{\mathsf{r}\tilde{\mathsf{r}}} \cdot e(g, a^{xyz_1(r_{id}+r_{id'})})^{\mathsf{r}\tilde{\mathsf{r}}} \cdot e(g, a^{xyz_2(s+|d|)})^{\mathsf{r}\tilde{\mathsf{r}}} \cdot e(g, a^{xyz_3(r_s+1)})^{\mathsf{r}\tilde{\mathsf{r}}} \cdot \\ &e(g, a^{xyz_4 L}) = e(g, a^{x+xy(id+id'+z_1(r_{id}+r_{id'})+z_2(s+|d|)+z_3(r_s+1)+z_4 L)})^{\mathsf{r}\tilde{\mathsf{r}}} = e(g, c'') \end{aligned}$$