# Two Secure Anonymous Proxy-based Data Storages*

Olivier Blazy[1], Xavier Bultel[2] and Pascal Lafourcade[2]
[1]*Université de Limoges, Xlim, Limoges, France*
[3]*Clermont Université Auvergne, LIMOS, BP 10448, 63000 Clermont-Ferrand, France*

Keywords: Anonymous storage; public key cryptography; proxy; standard model.

Abstract: Unidirectional proxy re-encryption (PRE) can be used to realize an efficient and secure shared storage. However, this type of storage does not yet protect its users' privacy: to retrieve some data a user must give his identity and his query to the proxy. We propose two secure data storage systems that allow authorized users to anonymously get access to the content of encrypted data on a storage. Each scheme corresponds to a certain economic model. In the first one, a user has to pay for each downloaded file, whereas in the second one, users pay each month a subscription to get an unlimited access to all their files.

## 1 Introduction

Users' anonymity is a property often required in modern security protocols especially in the post-Snowden era, not only towards outsiders but also towards intermediaries in a protocol. To motivate our work, we consider the following scenario where a video provider, called the *owner*, stores some videos on a database. To control access to the videos, he stores all of them encrypted on a remote server, and allows access to accredited users modulo some fees. Users wish to retrieve videos easily, and to disclose no information about their interest. According to some marketing issues the owner can enroll the users in different groups, like premium or privilege according to the prices of the services. The owner wants to meet the requirement of each group of users, and to bill them accordingly. He also wants an easy update procedure so that he can revoke access to users who misbehave and/or are late in the payment. Using a *proxy*, who helps the user to open encrypted videos in an anonymous way, we provide two solutions walking the fine line between these two seemingly opposite requirements (anonymity of the users and billing of the owner).

A naïve solution is to use a public key cryptosystem and an anonymous storage. The owner constructs a pair of public/private key for a group of users and gives the private key to each group member. Anybody can encrypt data with the public key and store it in the database. Thus, each group member can anonymously recover this encrypted data and retrieve the plaintext with the help of the group secret key. Unfortunately, this method does not allow the owner to revoke a group user, except by resetting all keys for groups and re-encrypting all the database where the revoked user belonged. This solution is not realistic.

Our goal is to design anonymous storage where each stored data is encrypted for a group of users and where the owner can easily revoke some users. Moreover, each authorized user can anonymously recover and decrypt data, *i.e.* without revealing neither his identity, nor his group, nor the recovered ciphertext. We also allow the owner of the storage, denoted $O$, to manage authorized users using two usage models: pay-per-download or monthly-fee. In the pay-per-download model, the owner only knows how many data are downloaded by a user and does not know which data have been asked by each user. The monthly-fee model is similar to Netflix[2], the user pays a subscription each month for an unlimited access to the data anonymously. The owner also manages the group of users: he can add new users or can revoke users that did not pay for this anonymous service. The revoked users cannot get access to any data anymore. To analyze the security of both systems, we consider an honest but curious proxy, also called *semi-trusted proxy*, who does not learn any information about the user and the requested files.

**Contributions:** We propose two schemes to anonymously manage a secure data storage. Like *proxy re-encryption* (PRE) based storage, the owner lets the access right management responsibility to the

*This research was conducted with the support of the "Digital Trust" Chair from the University of Auvergne Foundation.

[2]A provider of on-demand Internet streaming media

proxy and can be always offline during storage access. Both schemes use a semi-trusted proxy that gives only to authorized users the material to open encrypted data anonymously collected. The proxy does not know enough information to learn the identity of the users, this guarantees their anonymity. Furthermore, updating public parameters and proxy material, the owner can revoke any user. Both instantiations use public key encryption functions and a proxy. Both schemes are round optimal since user and proxy exchange in one round only. Our schemes differ on how the owner manages the users' rights.

- The first scheme, called *Direct Revocation Anonymous Storage* (DRAS), offers a pay-per-download model. The owner $O$ manages users such that: i) he knows how many files have been downloaded anonymously by each user ii) he revokes a user with a *direct* mechanism based on a black list BL sent by $O$ to the proxy $P$ and containing identities of users.
- The second one, called *Indirect Revocation Anonymous Storage* (IRAS) offers a monthly-fee model. The owner gives each month a key to each user who has paid his subscription to get an unlimited access to all files. The revocation mechanism is *indirect*, meaning that the owner only has to change his own public key when he updates each month and distributes cryptographic material for all remaining users.

We use an anonymous access to the information stored in a database $D$. This can be done using a Private Information Retrieval (Chor et al., 1995) that allows to retrieve a line in the database. This is more efficient than an Oblivious Transfer (Rabin, 1981) based solution, with a slight compromise on the obliviousness. In both cases, solutions exist to communicate with the proxy in only one round. We use such a mechanism to collect anonymously encrypted data in the database $D$. We will not focus on the kind of instantiation chosen as this is out of the scope of our contribution. However notice that pairing based solutions exist in the Standard Model with various level of security depending on the global security expected (Peikert et al., 2008, Abdalla et al., 2013, Blazy and Chevalier, 2015).

Moreover, for each scheme, we define the security models for anonymity and revocability. Then we prove that our two solutions are secure in the Standard Model, with respect to the model of semi-trusted proxy and under usual cryptographic assumptions. The models and the proofs are given in the full version of this paper. Our schemes have the following security properties:

- All data are IND-CPA encrypted for any unauthorized user, any revoked user and the proxy.
- DRAS scheme is partially anonymous, since the proxy $P$ can link a user to a group, only if the user uses several times one of his group's member key. Thanks to the information stored by the proxy the owner knows only the identity of user and the number of files that he has downloaded. Then a user has to pay twice when he downloads twice a file in order to preserve his privacy.
- The IRAS scheme is fully anonymous, it means that the proxy does not learn any information on the ciphertext, the user and his membership group. The owner can only recover the group who has requested the files. This scheme uses a *Smooth Projective Hash Function* (SPHF (Cramer and Shoup, 2002)) to be fully anonymous.

The required storage for our solutions is linear in the number of stored data times the number of groups for each data. Our protocols are also optimal in terms of communication costs.

**Related Work:** Shared storage based on *proxy re-encryption* (PRE) are very attractive for many reasons: after initialization of the proxy, the data owner is not required to manage the rights policy of the storage. Thus, the proxy can remain offline. To revoke a user's right, the owner simply updates informations given to the proxy. The owner can also give access to a set of data to several users without duplication of ciphertexts in the storage. However several disadvantages remain in PRE: partial trust in the proxy is mandatory, and there is no privacy protection. Indeed, the proxy learns several information about the user: his identity, his access rights and all ciphertexts that he wants to decrypt.

In (Blaze et al., 1998), Blaze *et al.* define "*proxy cryptography*". To delegate decryption rights they use a *semi-trusted* proxy, meaning that the proxy follows the protocol but tries to learn some information. The proxy receives the decryption rights, from a user $A$ to a user $B$, even when user $A$ is offline. The first bidirectional proxy re-encryption scheme has been described in (Ivan and Dodis, 2003): a user $A$ gives delegation rights to the proxy by forging a re-encryption key from both secret keys of $A$ and $B$. In (Ateniese et al., 2006), they describe the first unidirectional proxy re-encryption that allows to delegate decryption rights with delegator secret key and only delegate's public key. This paper introduces PRE schemes to manage an encrypted shared storage. Authors give an analysis of their PRE based storage implementation. In (Canetti and Hohenberger, 2007), they give a chosen ciphertext security definition of proxy re-encryption and provide a secure bidirectional scheme. In (Libert and Vergnaud, 2008),

Libert *et al.* present the first chosen ciphertext secure unidirectional proxy re-encryption. Some other results deal with anonymity and privacy protection in PRE based storage: in (Ateniese et al., 2009), authors present the notion of *key-private* (or *anonymous re-encryption key*) in PRE. This property does not allow the proxy to find the user public key corresponding to a re-encryption key. In (Zheng et al., 2014), authors propose a scheme that is both CCA secure and anonymous (in the anonymity model given by the previous article). On the other hand, the notion of *Anonymous proxy re-encryption*, from (Shao et al., 2012), protects the identity of the ciphertext (or re-encrypted ciphertext) recipient for anybody who does not know the re-encryption keys.

However, **all these cryptographic primitives only partially resolve the privacy protection problem** of PRE: the proxy can link all transactions of the same user, since the proxy uses the same re-encryption keys for each user. Moreover, ciphertexts that the proxy decrypts are not protected, and he can deduce the identity of the user. Our aim is to solve this problem by proposing two secure proxy based data storages that have strong privacy properties while trying to keep the benefits of PRE based storage (minimizing data duplication).

To control the access to the data, a classical solution would be to use an anonymous credential system (Chaum, 1985) where users are given credential corresponding to their access rights. In the following, the granularity of those credentials are set at group level. This allows us to define anonymity on two levels: first anonymity inside a group, even knowing which group can access a data, the user interacting with the protocol keeps his identity secret from an adversary. Such notion directly echoes to group signatures (Chaum and van Heyst, 1991), where anonymity is only defined for user belonging to the group. Second a weaker version of group anonymity: it should be difficult to learn which group a user belongs to.

## 2 Schemes Description

Our idea is to achieve anonymity of users and efficient user revocation mechanism by using proxy cryptography to manage users. To preserve the benefits of the anonymous storage, the proxy should not be able to learn any private information about the user (identity, access rights and data that he needs to decrypt).

From a high level point of view, our two schemes have a similar workflow: Firstly, the owner $O$ of the storage builds and distributes required keys for groups $G_j$ of users that have same access rights. Secondly, he generates a secret group member key $\mathsf{SKG}_j$ for each group and publishes the associated public key $\mathsf{PKG}_j$. Everybody can, using groups public key, store in the database $D$ an encrypted data $C$ for one of the groups. Lastly, the owner generates and distributes the secret key $\mathsf{SKP}$ to the proxy and a secret key $\mathsf{MSK}_i^j$ to each user $U_i$ belonging to the group $G_j$. Using these keys a user and the proxy interact, allowing user to decrypt the encrypted data $C$. Moreover the owner, who has all private keys, can decrypt all ciphertexts. In Fig. 1, we give the workflows of our two schemes that work as follows after the keys distribution in a setup phase:

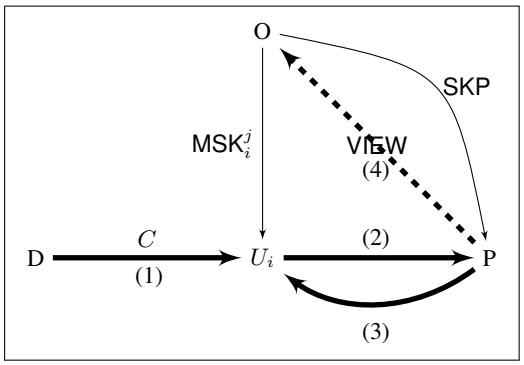


Figure 1: Schemes overview, where dashed line is only for DRAS, bold arrows represent decryption interactions and thin arrows correspond to the key's distribution (where SKP is the proxy secret key and $\mathsf{MSK}_i^j$ is the group member key for the user $U_i$ in the group $G_j$).

- Step (1): a user $U_i$ privately downloads a ciphertext $C$ from the anonymous storage $D$, using for instance a PIR. However, his member key $\mathsf{MSK}_i^j$ is not sufficient to open $C$.
- Step (2) and (3): the user $U_i$ interacts with the proxy in order to obtain the required material to open $C$. The proxy verifies the authorization of $U_i$ and only helps authorized users to open $C$.
- Step (4): in DRAS scheme, the proxy $P$ sends some information for the billing phase in the message denoted by VIEW to the owner $O$.

We describe our two schemes starting with the DRAS scheme as a stepping stone for our IRAS scheme. On one hand, DRAS scheme has the following revocation property: the owner can easily revoke any user at any time. However this scheme does not perfectly preserve the privacy of users: if a user requests the proxy twice with two data encrypted with the same group key, then the proxy can link the two requests. On the other hand, IRAS scheme is more complex and offers a different revocation mechanism since we cannot revoke the users at any time, but this scheme allows us to perfectly preserve all private information of the users.

## 2.1 Description of DRAS Scheme

We use an IND-CCA2 public key cryptosystem $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Anybody can encrypt a data with the public key of the group $\mathsf{PKG}_j$ and store it in an anonymous database $D$. This encryption is done with ElGamal based (ElGamal, 1985) encryption denoted by $\mathsf{Encrypt}(\mathsf{PKG}_j, m)$, where $m$ is the data and $\mathsf{PKG}_j$ the group public key. The main idea behind DRAS is to allow the owner to revoke a user using a blacklist BL transferred by $O$ to the proxy $P$ and to use a white list WL to bill the authorized users. The white list is composed of couples of identities and re-encryption keys. With the blacklist, $P$ is able to help users that are not on BL to open the ciphertexts.

In Fig. 2, we describe DRAS that contains:

$\mathsf{Setup}(\mathfrak{K})$: take a security parameter $\mathfrak{K}$ as input and return the parameters $\mathcal{P} = (g, \mathbb{G}, p, \mathsf{PKE})$.

$\mathsf{P\text{-}Gen}(\mathcal{P})$: generate the pair of keys $(\mathsf{PKP}, \mathsf{SKP})$ for the proxy $P$,

$\mathsf{G\text{-}Gen}(\mathcal{P})$: generate the pair of keys $(\mathsf{PKG}_j, \mathsf{SKG}_j)$ for the group of users $G_j$,

$\mathsf{Join}(\mathsf{SKG}_j, \mathsf{WL}, U_i)$: generate the secret key $\mathsf{MSK}_i^j$ that allows a user $U_i$ of a group $G_j$ to transform a ciphertext $C$ to the proxy. The user identity $U_i$ and the re-encryption key $\mathsf{MSK}_i^j$ are added to the white list WL.

$\mathsf{Encrypt}(\mathsf{PKG}_j, m)$: allow a user to encrypt the message $m$ for the group $G_j$ in $C$ and to store it in the database $D$.

Those algorithms allow the owner to set up properly the system. The first ones generate all the required keys, while the last one allows anyone to add a data to the database encrypted for a specific group. To decrypt a message, there are two different ways:

$\mathsf{Decrypt}(\mathsf{SKG}_j, C)$: use a group secret key $\mathsf{SKG}_j$ to recover the plaintext in $C$.

$\mathsf{ProxyDec}(U_i, P)$: a protocol between a user $U_i$ and the proxy $P$ allowing a non revocated user to interact with the proxy, in order to decrypt the ciphertext $C$. The proxy outputs to the owner $O$ the value VIEW that contains data needed for billing.

Those algorithms allow a user to recover an encrypted data stored in the database $D$. The first one allows the owner to decrypt any data of $D$, while the second one allows a user $U_i$ to access an encrypted message, and decrypts it through an exchange protocol with the proxy that is described in Fig. 3. First, user encrypts, with the public key PKP, a randomization of $C_1$ (the first part of the ciphertext $C$) and RK (the first part of $\mathsf{MSK}_i^j$). Once the message $B$ is received by the proxy. After verification that the key RK is not blacklisted ($\tilde{B}_2 \notin$ BL), the proxy, using RK and SKP (a key unknown by the user), transforms the part $\tilde{B}_1$ into another ciphertext $\hat{C}$. The user $U_i$ then recovers the message $m$ using his secrets. On the same time the proxy provides a VIEW of the interaction to the owner. Finally, DRAS contains:

$\mathsf{Revoke}(\mathsf{MSK}_i^j, \mathsf{BL})$: add the user $U_i$ using $\mathsf{MSK}_i^j$ to the blacklist BL of revocated users.

$\mathsf{Open}(\mathsf{VIEW}, \mathsf{WL})$: output the user identity corresponding to the VIEW message.

Those algorithms are for administrative purposes. The first one, simply allows to revoke a user, without having to update the whole database, while the second one allows the owner to exploit the VIEW in order to bill according the number of data downloaded.

## 2.2 Description of IRAS scheme

Under the monthly-fee model, a revocation system at any time is not usefull anymore. Then the blacklist and the white list used for billing and for revocation are not required anymore. We propose IRAS scheme, that has a different revocation process: the owner revokes a user by updating his own signing key. Once an owner's key is updated, all group member keys are unusable as long as the owner updates them. Morover, this scheme is perfectly anonymous for any users, and all decryptions are not traceable by $O$. Comparing to the previous scheme, IRAS is more complex: it requires an EUF-CMA signature scheme $\mathcal{S} = (\mathsf{Gen}_{\mathcal{S}}, \mathsf{Sign}_{\mathcal{S}}, \mathsf{Verif}_{\mathcal{S}})$, a bilinear pairing of type 3, an IND-CPA public key cryptosystem $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and a Smooth Projective Hash Function (SPHF). We start by recalling how an SPHF works before presenting IRAS.

*Smooth projective hash functions* (SPHF) were introduced by Cramer *et al.* to construct encryption schemes in (Cramer and Shoup, 2002). A projective hashing family is a family of hash functions that can be evaluated in two ways: either using the (secret) *hashing* key, one can compute the function on every point in its domain, or using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found numerous applications in various contexts in cryptography (*e.g.* (Gennaro and Lindell, 2003, Kalai, 2005, Abdalla et al., 2009, Blazy et al., 2012)).

**Definition 1** (Smooth Projective Hashing System)**.** *A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$ and $W \in \mathfrak{L}$, with hashes $v, v' \in \mathcal{V}$ is defined by the following algorithms:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *generates the global parameters $\mathcal{P}$ of the scheme, and the description of an $\mathcal{NP}$ lan-*

$\mathsf{Setup}(\mathfrak{K})$: pick a group $\mathbb{G}$ of prime order $p$ generated by $g$, returns $\mathcal{P} = (g, \mathbb{G}, p, \mathsf{PKE})$.

$\mathsf{P\text{-}Gen}(\mathcal{P})$: use $\mathsf{Gen}$ to generate $(\mathsf{PKP}, \mathsf{SKP})$.

$\mathsf{G\text{-}Gen}(\mathcal{P})$: pick $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$, set $\mathsf{PKG}_j = g^\gamma$ and $\mathsf{SKG}_j = \gamma$, and then return $(\mathsf{PKG}_j, \mathsf{SKG}_j)$.

$\mathsf{Join}(\mathsf{SKG}_j, \mathsf{WL}, U_i)$: For $\mathsf{SKG}_j = \gamma$, pick $t \xleftarrow{\$} \mathbb{Z}_p^*$, set $\mathsf{MSK}_i^j = (t, \mathsf{RK})$ where $\mathsf{RK} = \mathsf{Enc}_{\mathsf{PKP}}(\frac{t}{\gamma})$, add $(\mathsf{MSK}_i^j, U_i)$ in $\mathsf{WL}$ and return $\mathsf{MSK}_i^j$.

$\mathsf{Encrypt}(\mathsf{PKG}_j, m)$: For $\mathsf{PKG}_j = g^\gamma$, user $U$ picks $r \xleftarrow{\$} \mathbb{Z}_p^*$, computes $C = (C_1, C_2) = (g^{\gamma r}, g^r \cdot m)$, and then stores $C$ in the database $D$.

$\mathsf{Revoke}(\mathsf{MSK}_i^j, \mathsf{BL})$: add $\mathsf{RK}$ to the blacklist $\mathsf{BL}$ where $\mathsf{MSK}_i^j = (t, \mathsf{RK})$.

$\mathsf{Open}(\mathsf{VIEW}, \mathsf{WL})$: The owner finds $U_i$ and $t$ such that $((t, \mathsf{VIEW}), U_i) \in \mathsf{WL}$ to get the $iU_i$ user.

$\mathsf{Decrypt}(\mathsf{SKG}_i^j, C)$: For $\mathsf{SKG}_i^j = \gamma$, return $m = \frac{C_2}{C_1^{1/\gamma}}$, where $C = (C_1, C_2)$.

$\mathsf{ProxyDec}(U_i, P)$: protocol described in Figure 3.

Figure 2: DRAS scheme.

| **User** $U_i$ | | **Proxy** $P$ |
|---|---|---|
| $\mathsf{PKP}; \mathsf{MSK}_i^j = (t, \mathsf{RK}); C = (C_1, C_2)$ | | $\mathsf{SKP}; \mathsf{BL}$ |
| $s \xleftarrow{\$} \mathbb{Z}_p^*$; $B = \mathsf{Enc}_{\mathsf{PKP}}((C_1)^s, \mathsf{RK})$ | $\xrightarrow{B}$ | $\widetilde{B} = (\widetilde{B}_1, \widetilde{B}_2) = \mathsf{Dec}_{\mathsf{SKP}}(B)$ |
| | | If $\widetilde{B}_2 \in \mathsf{BL}$ then abort; else $w = \mathsf{Dec}_{\mathsf{SKP}}(\widetilde{B}_2)$ |
| $m = \frac{C_2}{\widehat{C}^{(1/st)}}$ | $\xleftarrow{\widehat{C}}$ | $\widehat{C} = (\widetilde{B}_1)^w$ |
| Output $m$ | | Output $\mathsf{VIEW} = \widetilde{B}_2$. |

Figure 3: $\mathsf{ProxyDec}(U_i, P)$ protocol for DRAS.

*guage $\mathfrak{L}$;*

- $\mathsf{HashKG}(\mathfrak{L}, \mathcal{P})$*, outputs a hashing key* $\mathsf{hk}$ *for the language $\mathfrak{L}$;*
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W)$*, derives the projection key* $\mathsf{hp}$*, thanks to the key* $\mathsf{hk}$*,*
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W)$*, outputs the hash value $v \in \mathcal{V}$, thanks to the key* $\mathsf{hk}$*, and $W$,*
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathcal{P}), W, w)$*, outputs the hash value $v' \in \mathcal{V}$, thanks to* $\mathsf{hp}$*,$W$ and the witness $w$.*

In the following, we consider $\mathfrak{L}$ as a hard-partitioned subset of $X$, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$. An SPHF should satisfy the following properties:

- *Correctness*: Let $W \in \mathfrak{L}$ and $w$ a witness of this membership. Then, for all hashing keys $\mathsf{hk}$ and associated projection keys $\mathsf{hp}$ we have: $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathcal{P}), W, w)$
- *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the distributions $\Delta_0$ and $\Delta_1$ are statistically indistinguishable: $\Delta_0 = \{(\mathfrak{L}, \mathcal{P}, W, \mathsf{hp}, v) |\ \mathcal{P} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathcal{P}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W), v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W)\}$ and $\Delta_1 = \{(\mathfrak{L}, \mathcal{P}, W, \mathsf{hp}, v) |\ \mathcal{P} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathcal{P}), \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathcal{P}), W), v \xleftarrow{\$} V\}$
- *Pseudo-Randomness*: If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable.

Our IRAS scheme is given in Fig. 4 and is composed of the following algorithms:

$\mathsf{Setup}(\mathfrak{K})$: takes a security parameter $\mathfrak{K}$ in input and return the parameters $\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, p, \mathsf{PKE}, \mathcal{S})$.

$\mathsf{O\text{-}Gen}(\mathcal{P})$: create the owner keys $(\mathsf{PKO}, \mathsf{SKO})$.

$\mathsf{P\text{-}Gen}(\mathcal{P})$: create the proxy keys $(\mathsf{PKP}, \mathsf{SKP})$.

$\mathsf{G\text{-}Gen}(\mathcal{P})$: generate the group keys $(\mathsf{PKG}_j, \mathsf{SKG}_j)$.

$\mathsf{Join}(\mathsf{SKG}_j, \mathsf{ssk_o}, \mathsf{PKP})$: compute a group member secret key $\mathsf{MSK}_i^j$ for the user $U_i$.

$\mathsf{O\text{-}Update}(\mathsf{SKO}, \mathsf{PKO})$: update the owner keys.

$\mathsf{U\text{-}Update}(\mathsf{MSK}_i^j, \mathsf{SKG}_j, \mathsf{SKO})$: update a user group member secret key $\mathsf{MSK}_i^j$.

$\mathsf{Encrypt}(\mathsf{PKG}_j, m)$: encrypt a message $m$ for a group into the ciphertext $C$.

$\mathsf{Decrypt}(\mathsf{SKG}_j, C)$: decrypt a ciphertext $C$ into $m$ using a group secret key $\mathsf{SKG}_j$.

$\mathsf{ProxyDec}(U_i, P)$: a protocol between a user $U$ and the proxy $P$, that allows an authorized user to recover the plaintext in $C$.

To revoke a user $U_i$ in a group $j$, the owner only updates his signature keys $(\mathsf{PKO}, \mathsf{SKO})$ used to sign the users member group key $\mathsf{MSK}_i^j$ and he does not re-sign it with his new signing key.

Moreover, each user has a secret key $\mathsf{MSK}_i^j$ for each of their groups. This key contains a re-encryption key $\mathsf{RK}$ encrypted with the public key of the proxy $\mathsf{PKP}$ by an ElGamal cryptosystem. Thus, a user can raise the re-encryption key $\mathsf{RK}$ to a random power $\mu$ in this ElGamal ciphertext thanks to the malleability property of this encryption scheme. This

$\mathsf{Setup}(\mathfrak{K})$: pick a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order $p$ and generators $g_1, g_2$ respectively with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and return $\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, p, \mathsf{PKE}, \mathcal{S})$.

$\mathsf{O\text{-}Gen}(\mathcal{P})$: generate a pair of signature keys $(\mathsf{ssk_o}, \mathsf{svk_o}) \leftarrow \mathsf{Gen}_{\mathcal{S}}$, a pair of encryption keys $(\mathsf{pk_o}, \mathsf{sk_o}) \leftarrow \mathsf{Gen}$ and return $\mathsf{PKO} = (\mathsf{pk_o}, \mathsf{svk_o})$ and $\mathsf{SKO} = (\mathsf{sk_o}, \mathsf{ssk_o})$.

$\mathsf{P\text{-}Gen}(\mathcal{P})$: generate $(\mathsf{pk_p}, \mathsf{sk_p}) \leftarrow \mathsf{Gen}$, pick $x \xleftarrow{\$} \mathbb{Z}_p^*$, set $\mathsf{PKP} = (\mathsf{pk_p}, g_2^x)$ and $\mathsf{SKP} = (\mathsf{sk_p}, x)$ and return $(\mathsf{PKP}, \mathsf{SKP})$.

$\mathsf{G\text{-}Gen}(\mathcal{P})$: pick $\gamma_0, \gamma_1 \xleftarrow{\$} \mathbb{Z}_p^*$, set $\mathsf{PKG}_j = (g_1^{\gamma_0}, g_1^{\gamma_1})$ and $\mathsf{SKG}_j = (\gamma_0, \gamma_1)$, and return $(\mathsf{PKG}_j, \mathsf{SKG}_j)$.

$\mathsf{Join}(\mathsf{SKG}_j, \mathsf{ssk_o}, \mathsf{PKP})$: pick $r_0, r_1 \xleftarrow{\$} \mathbb{Z}_p^*$, knowing that $\mathsf{SKG} = (\gamma_0, \gamma_1)$, set $\mathsf{RK} = (\mathsf{RK}_1, \mathsf{RK}_2, \mathsf{RK}_3, \mathsf{RK}_4)$ where $\gamma' = \frac{\gamma_1}{\gamma_0}$, $\mathsf{RK}_1 = g_2^{r_0}$, $\mathsf{RK}_2 = g_2^{xr_0} \cdot g_2^{(1/\gamma_0)}$, $\mathsf{RK}_3 = g_2^{r_1}$, $\mathsf{RK}_4 = g_2^{xr_1} \cdot g_2^{\gamma'}$ and $\sigma = \mathsf{Sign}_{\mathcal{S}}(\mathsf{ssk_o}, \mathsf{RK})$ and then return $\mathsf{MSK}_i^j = (\mathsf{RK}, \sigma)$.

$\mathsf{O\text{-}Update}(\mathsf{SKO}, \mathsf{PKO})$: generate a fresh $(\mathsf{ssk'_o}, \mathsf{svk'_o}) \leftarrow \mathsf{Gen}_{\mathcal{S}}$ and update $\mathsf{svk_o} := \mathsf{svk'_o}$ and $\mathsf{ssk_o} := \mathsf{ssk'_o}$ in $\mathsf{SKO} = (\mathsf{sk_o}, \mathsf{ssk_o})$ and $\mathsf{PKO} = (\mathsf{pk_o}, \mathsf{svk_o})$.

$\mathsf{U\text{-}Update}(\mathsf{MSK}_i^j, \mathsf{SKO})$: compute $\mathsf{MSK}_i^{j'} = (\mathsf{RK}, \mathsf{Sign}_{\mathcal{S}}(\mathsf{ssk_o}, \mathsf{RK}))$ from $\mathsf{MSK}_i^j = (\mathsf{RK}, \sigma)$ and $\mathsf{SKO} = (\mathsf{sk_o}, \mathsf{ssk_o})$. Update $\mathsf{MSK}_i^j := \mathsf{MSK}_i^{j'}$.

$\mathsf{Encrypt}(\mathsf{PKG}, m)$: pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, compute $C = (g_1^{\gamma_0 s}, g_1^{\gamma_1 s}, e(g_1, g_2)^s \cdot m)$, where $\mathsf{PKG} = (g_1^{\gamma_0}, g_1^{\gamma_1})$ and after storing $C$ in the database $D$ return it.

$\mathsf{Decrypt}(\mathsf{SKG}_j, C)$: return $m = \frac{C_3}{e(C_1, g_2)^{1/\gamma_0}}$ using $\mathsf{SKG}_j = (\gamma_0, \gamma_1)$ and $C = (C_1, C_2, C_3)$.

$\mathsf{ProxyDec}(U_i, P)$: protocol described in Figure 5. Following our SPHF: the proxy computes the hash keys $\mathsf{hk} = \mathsf{HashKG}(\widetilde{\mathsf{RK}}, E)$ for the language stating that $\widetilde{\mathsf{RK}}$ is a valid randomization of the value encrypted in $E$, signed by the authority in the committed $\sigma$ using PKO. If the user $U_i$ is revoked, this value will be indistinguishable from a random noise. This gives him $\mathsf{hp} = \mathsf{ProjKG}(\widetilde{\mathsf{RK}}, E, \mathsf{hk})$ and a hash value $H = \mathsf{Hash}(\widetilde{\mathsf{RK}}, E, \mathsf{hk})$.

Figure 4: IRAS Scheme.

allows the user to send to the proxy a re-encryption key $\mathsf{RK}'$ which is RK hidden by a secret value $\mu$. The proxy, using the ElGamal secret key SKP, finds the masked re-encryption key $\mathsf{RK}'$ to compute the information $\widehat{C}$ which is also hidden by the same secret values $\mu$ that hides the re-encryption key RK. This masked value $\widehat{C}$ gives no information to the proxy who sends it to $U_i$. The user can remove the mask $\mu$ on $\widehat{C}$ and obtain the information that he needs to compute the plaintext $m$. This decryption protocol is called $\mathsf{ProxyDec}(U_i, P)$ and is detailed in Figure 5. It is a 3-step protocol:

- First the user, sends two ciphertexts $B$ and $E$ to the proxy. $B$ is an encryption for the proxy, which contains a randomization of the ciphertext $C$ and a randomization of the re-encryption key RK. Moreover, $E$ is an encryption of the user group member secret key $\mathsf{MSK}_i^j$ for the owner.
- Then the proxy decrypts the values in $B$, checks the consistency of the various randomizations, and if this holds, the proxy computes $\hat{C}_0$ a decryption of the randomized ciphertext chosen by the user. Now, since the full decryption should only be done for truthful user, the proxy hides the outcome of the decryption. He does it with the hash value associated with the following language of the SPHF: $E$ is an encryption of an $\mathsf{MSK}_i^j$ signed by the owner, and is consistent with the randomized values given in $B$ (namely there exists a $\mu$ linking the encrypted $\mathsf{RK}_1, \mathsf{RK}_2$ with $\mathsf{RK}_1^\mu, \mathsf{RK}_2^\mu$, and a $\mu'$ linking the encrypted $\mathsf{RK}_3, \mathsf{RK}_4$ with $\mathsf{RK}_3^{\mu'}, \mathsf{RK}_4^{\mu'}$). To do so, the proxy computes a hash key $\mathsf{hk} = \mathsf{HashKG}(\widetilde{B}_{\mathsf{MSK}}, E)$, its associated projected key $\mathsf{hp} = \mathsf{ProjKG}(\widetilde{B}_{\mathsf{MSK}}, E, \mathsf{hk})$, and computes the associated hash $H = \mathsf{Hash}(\widetilde{B}_{\mathsf{MSK}}, E, \mathsf{hk})$. Those algorithms are derived from those on the sublanguages as described afterwards.
  The proxy then simply uses $H$ to do a one time pad of $\hat{C}_0$ and sends the resulting value together with the projected key hp to the user. The Smoothness of the SPHF ensures that if the user misbehaved, this will be indistinguishable from a random noise. (This step allows the user to do an implicit proof of knowledge of his secret key encrypted under PKO)
- Now the user, knowing the randoms he used to generate $E$ and to do the initial randomization $(\mu, \mu')$, is able to compute the Projected Hash $H' = \mathsf{ProjHash}(\mathsf{hp}, \mu, \mu')$. If he is honest then $H = H'$, so he can recover $\hat{C}_0$, from which he recovers the message $m$.

**Constructing the required Smooth Projective Hash Function:** Conjunction techniques were detailed in (Abdalla et al., 2009), to show how to combine Smooth Projective Hash Functions on various languages. In the following, we consider the various building blocks required by the language above. One side, we need to consider the language of valid

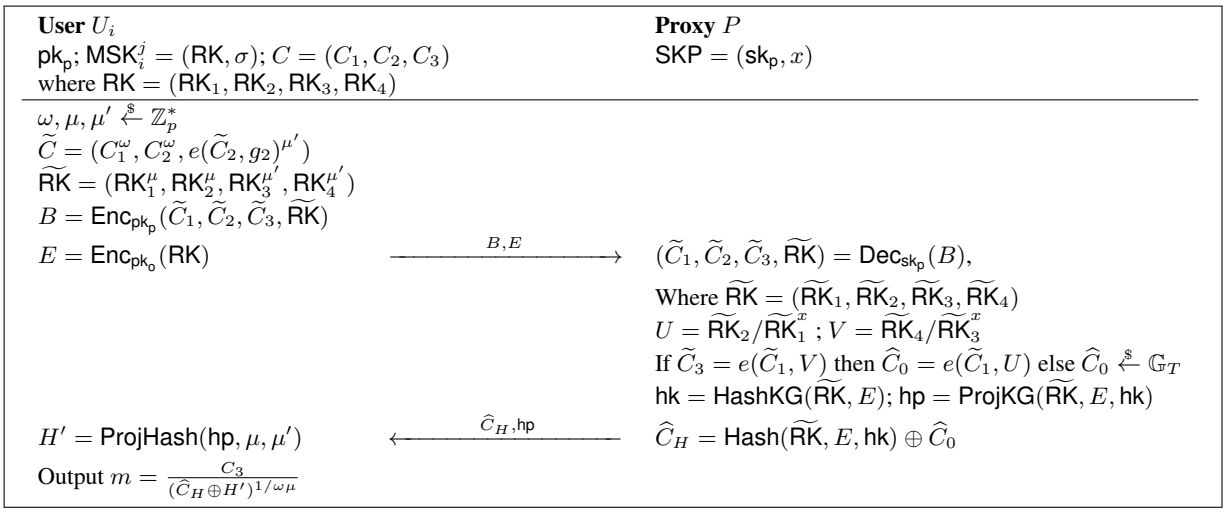


Figure 5: $\mathsf{ProxyDec}(U_i, P)$ protocol for IRAS.

randomization of encrypted values, on the other we consider the language of the encryption of a valid signature under a given public key.

- The first language requires to show (twice), that a pair of values are a proper randomization of an encrypted value. We then suppose that there exists an ElGamal encryption scheme described by the generators $h$ and $g$. A user possesses non trivial values $X, Y$ *i.e.* $X = g^x, Y = g^y$ for $x \neq 0, y \neq 0$, and wants to send $(h^r X, g^r, h^s Y, g^s)$ the ElGamal encryption of $X, Y$ (with $h^r X \neq 1_{\mathbb{G}}$), together with a randomization of $X^a, Y^a$. The proxy wants to build a smooth projective hash function for the language $\mathfrak{L}$ of following tuple: $W = (h^r X, g^r, h^s Y, g^s, X^a, Y^a) = (W_1, W_2, W_3, W_4, W_5, W_6)$.
  - $\mathsf{HashKG}(\mathfrak{L}, \mathcal{P})$, picks random scalars $\lambda, \eta, \theta, \mu \xleftarrow{\$} \mathbb{Z}_p^*$ to define $\mathsf{hk}_1$,
  - $\mathsf{ProjKG}(\mathsf{hk}_1, (\mathfrak{L}, \mathcal{P}), W)$, derives $\mathsf{hp}_1 = W_1^\lambda W_3^\mu W_2^\eta W_4^\theta, h^\lambda g^\eta, h^\mu g^\theta$,
  - $\mathsf{Hash}(\mathsf{hk}_1, (\mathfrak{L}, \mathcal{P}), W)$, outputs a hash value $v = W_5^\lambda W_6^\mu$, thanks to $\mathsf{hk}_1$, and $W$,
  - $\mathsf{ProjHash}(\mathsf{hp}_1, (\mathfrak{L}, \mathcal{P}), W, w)$, thanks to $\mathsf{hp}_1$ and the witness $w$ that $W \in \mathfrak{L}$, outputs the hash value $v' = \left(\mathsf{hp}_{1,1} \mathsf{hp}_{1,2}^{-r} \mathsf{hp}_{1,3}^{-s}\right)^a$.
- The second language requires to show that we have an encryption of a valid signature, as in (Blazy et al., 2012) by combining a ElGamal Encryption and an asymmetric Waters signature, denoted $\mathsf{ek} = g_1^y$ and $\mathsf{vk} = g_2^z$.
  - $\mathsf{HashKG}(\mathcal{L})$, picks $x_1, x_2 \in \mathbb{Z}_p^*$ to define $\mathsf{hk}_2$;
  - $\mathsf{Hash}(\mathsf{hk}_2; \mathcal{L}, C)$, outputs a hash value $H = e(c_1, g_2)^{x_1} \cdot (e(c_2, g_2)/(e(h_1, \mathsf{vk}) \cdot e(F(M), \sigma_3)))^{x_2}$;
  - $\mathsf{ProjKG}(\mathsf{hk}_2; \mathcal{L}, C)$ derives $\mathsf{hp}_2 = g_1^{x_1} \mathsf{ek}^{x_2}$;
  - $\mathsf{ProjHash}(\mathsf{hp}_2; \mathcal{L}, r_1)$ computes the projected hash value $e(\mathsf{hp}_2^{r_1}, g_2)$.

One can further extend the language to prove that one possesses a valid signature under a public verification key $\mathsf{vk}$ on an element in this language.

IRAS requires to prove that two pairs are valid randomization, hence using conjunction techniques from (Abdalla et al., 2009) one obtains a projection key $\mathsf{hp}$ involving six elements for the pairs (two iterations of $\mathsf{hp}_1$, one for the randomization of the pair $(\mathsf{RK}_1, \mathsf{RK}_2)$ with $\mu$, and one for the randomization of $(\mathsf{RK}_3, \mathsf{RK}_4)$ with $\mu'$) and an additional five elements for the projected keys to handle the signature on an encrypted message (one $\mathsf{hp}_2$ for the language of valid encrypted signature $\sigma$, and four to combine with the encrypted message $\mathsf{RK}$).

## 3 Conclusion

We have presented two secure anonymous storages based on proxy cryptography. The aim of the proxy is to allow only authorized users to open ciphertexts collected anonymously. We proposed two schemes answering to different use cases. The first one, called DRAS, allows the owner to revoke a user by giving a black list to the proxy, while the second one, called IRAS, provides an indirect way for the owner to revoke a user, by not renewing his key. Then only the users who have paid their subscriptions receive the new keys and can open the encrypted files. Those constructions are proved secure in the Standard Model while considering a semi-trusted proxy.

The next step will be to propose a concrete implementation of our two schemes in order to com-

pare their efficiency. Moreover, one can see that in some of our proofs, the proxy can even be a little more malicious than just curious without impacting the user privacy, and it would be a major development to construct a resilient scheme in the covert adversary model. In (Fiat and Naor, 1994) Broadcast Encryption (BE) was introduced, current BE schemes allow to send messages to a given set of users depending on their current subscription/privileges and the revocation is done in a similar way to our second scheme, by updating the decryption keys at certain time period. However this requires a user to be online all the time and prevents on the fly access to stored messages. We would like to find a nice combination of these two techniques.

## References


Abdalla, M., Benhamouda, F., Blazy, O., Chevalier, C., and Pointcheval, D. (2013). SPHF-friendly non-interactive commitments. In Sako, K. and Sarkar, P., editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer.

Abdalla, M., Chevalier, C., and Pointcheval, D. (2009). Smooth projective hashing for conditionally extractable commitments. In Halevi, S., editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer.

Ateniese, G., Benson, K., and Hohenberger, S. (2009). Key-private proxy re-encryption. In Fischlin, M., editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 279–294. Springer.

Ateniese, G., Fu, K., Green, M., and Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30.

Blaze, M., Bleumer, G., and Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In Nyberg, K., editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 127–144. Springer.

Blazy, O. and Chevalier, C. (2015). Generic construction of UC-secure oblivious transfer. In *ACNS 15*, LNCS, pages 65–86. Springer.

Blazy, O., Pointcheval, D., and Vergnaud, D. (2012). Round-optimal privacy-preserving protocols with smooth projective hash functions. In Cramer, R., editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer.

Canetti, R. and Hohenberger, S. (2007). Chosen-ciphertext secure proxy re-encryption. In Ning, P., di Vimercati, S. D. C., and Syverson, P. F., editors, *ACM CCS 07*, pages 185–194. ACM Press.

Chaum, D. (1985). Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044.

Chaum, D. and van Heyst, E. (1991). Group signatures. In Davies, D. W., editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer.

Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. (1995). Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press.

Cramer, R. and Shoup, V. (2002). Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Knudsen, L. R., editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472.

Fiat, A. and Naor, M. (1994). Broadcast encryption. In Stinson, D. R., editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer.

Gennaro, R. and Lindell, Y. (2003). A framework for password-based authenticated key exchange. In Biham, E., editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer.

Ivan, A. and Dodis, Y. (2003). Proxy cryptography revisited. In *NDSS 2003*. The Internet Society.

Kalai, Y. T. (2005). Smooth projective hashing and two-message oblivious transfer. In Cramer, R., editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer.

Libert, B. and Vergnaud, D. (2008). Unidirectional chosen-ciphertext secure proxy re-encryption. In Cramer, R., editor, *PKC 2008*, volume 4939 of *LNCS*, pages 360–379. Springer.

Peikert, C., Vaikuntanathan, V., and Waters, B. (2008). A framework for efficient and composable oblivious transfer. In Wagner, D., editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer.

Rabin, M. O. (1981). How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University.

Shao, J., Liu, P., Wei, G., and Ling, Y. (2012). Anonymous proxy re-encryption. In *Security Comm. Networks, 5: 439–449. doi: 10.1002/sec.326*.

Zheng, Q., Zhu, W., Zhu, J., and Zhang, X. (2014). Improved anonymous proxy re-encryption with CCA security. In Moriai, S., Jaeger, T., and Sakurai, K., editors, *ASIACCS 14*, pages 249–258. ACM Press.