Secure Cloud Storage with Encrypted Data using File Based Authentication

Jia Xu
National University of Singapore
jiaxu2001@gmail.com

Ee-Chien Chang
National University of Singapore
changec@comp.nus.edu.sg

Jianying Zhou
Institute for Infocomm Research
jyzhou@i2r.a-star.edu.sg

Abstract

Cloud storage service is gaining popularity in recent years. Client-side deduplication is widely adopted by cloud storage services like Dropbox and MozyHome, to save bandwidth and storage. Security flaws, which may lead to private data leakage, in the current client-side deduplication mechanism are found recently by Harnik, Pinkas, and Shulman-Peleg (S&P Magazine, '10).

This paper presents a notion of File Based Authentication*, that is, a user authenticates himself/herself using the file he/she possesses as the secret information. File Based Authentication can be applied to protect confidentiality of users' sensitive data file in the cloud storage from both the third party attackers and the semi-honest cloud server itself. The proposed solution enables efficient client-side deduplication (across users) and client-side sharing of encrypted data in the cloud storage, where the encryption key is chosen by a user* independently *(without negotiation with other owners of the same data file) and is kept secret from the cloud storage server and its client software.*

Furthermore, the surprising part is that, a secure cloud storage system without pre-registration for users can be constructed based on the proposed solution: All owners of a file F share an a priori account with hash value $h(F)$ as identity and sensitive file F as password, without knowledge of each other and without registration on the cloud storage server, since the first uploading of F (in encrypted form) to the cloud will function as the registration of account $(h(F), F)$ in the cloud storage server.

Keywords

Privacy, Cloud Storage, Client-side Deduplication, Proofs of Ownership, Client-side Sharing, File Based Authentication, Registration-free Cloud Service

1. Introduction

Cloud storage service is gaining popularity. To reduce resource consumption in bandwidth and storage, many cloud storage services including Dropbox [16] employs client-side deduplication [32]. That is, when a user tries to upload a file to the server, the server checks whether this particular file is already in the cloud and saves the uploading process if it is. In this way, every single file will have only one copy in the cloud. Recently, an attack on current client-side mechanism in popular cloud storage service like Dropbox is proposed [19], [18]: If the adversary *somehow* has the hash value of a file stored in the cloud storage, he could fool the cloud server that he has the file by presenting only the hash value in the client-side deduplication process, and thus gain access to that file in the cloud.

Halevi *et al.* [18] proposed a primitive called *Proofs of Ownership* (PoW) to allow an owner of a file to prove to the server that he/she indeed has that file in an efficient way. Informally, in a Proofs of Ownership scheme, if an adversary *somehow* has some limited knowledge on a file, he cannot gain significantly more knowledge of that file via the client-side deduplication process with the cloud storage server. Shortly, information leakage should not be amplified in the client side deduplication process.

Proofs of Ownership alone achieves strong security. However, when deployed in a cloud storage service, the security of the overall system is limited by other components, like user authentication system, of the cloud service. In a practical cloud storage service secured by PoW [18], adversary still can obtain the large data file from a short secret value—the user name and password of (one of) the owner of that file. Furthermore, the root account password of the cloud server can be considered as a short "representation" of all files stored in the cloud storage. Once an adversary *somehow* obtains the root password, he can access every single file in the cloud, which is a huge amplification of information leakage.

Furthermore, all Proofs of Ownership schemes proposed in Halevi *et al.* [18] have to trust the cloud server and only protect the data confidentiality from third party attackers, since these schemes cannot support encrypted data files. However, sensitive data files are

likely to be encrypted on the client side by users before uploaded to the cloud storage.

Different from PoW [18], this paper takes the user authentication system into account, and aims to protect confidentiality of users' data in the cloud storage against both the third party attackers and the semi-honest cloud server itself, where a semi-honest cloud server is curious about the content of users' sensitive data but will guarantee the integrity and availability of users' data. If the cloud server is semi-honest, our solution achieves higher level of security, compared to cloud storage system secured with PoW; if the cloud server is honest, our solution achieves the same level of security. Particularly, in our solution, there is no single short secret value, which can "represent" all files in the cloud. Furthermore, our solution can be incorporated with PoW and Proofs of Storage (POR [21], [28] and PDP [2]) in a seamless way.

It is known that cloud storage service SpiderOak [29] provides some sort of data deduplication over encrypted data in the cloud. However, in SpiderOak service, users' data files are encrypted on the client side by SpiderOak client software which is closed source, and apparently the encryption key is chosen by the SpiderOak client software. In contrast, in this paper, a user's data file is encrypted using a publicly available encryption algorithm with the encryption key chosen by the user independently, without any negotiation with other owners of the same file or the cloud server. We emphasize that the encryption key should be kept safely from the cloud server (including the client software of the cloud service).

1.1. Our results and Contributions

Our results and contributions in this paper are summarized as below.

1.1.1. File Based Authentication. This paper proposes a notion of *File Based Authentication*. In a File Based Authentication system, an owner of a file F can authenticate himself/herself to the cloud server, using the hash value $h(F)$ as identity and the sensitive file F as the password, with a zero-knowledge proof. All owners of F share the a priori account $(h(F), F)$, although they may have no knowledge of each other.

We formalize File Based Authentication, and propose a construction. We prove the security of proposed construction in general case in random oracle model, and in simple case in standard model. Particularly, we employ an almost universal hash family to replace the random oracle, and the leftover hash lemma will help us complete the proof in the standard model.

1.1.2. Client-side Deduplication across users with Encrypted Data. To apply File Based Authentication in cloud storage, each data file F will be encrypt using a short one-time key τ_F called trapdoor and this trapdoor will be encrypted using the file F as secret key. Next, the resulting ciphertexts of the data file and the trapdoor will be uploaded to the cloud storage—This uploading functions as registration of account $(h(F), F)$ to the cloud storage server.

If an owner, say Alice, of F tries to authenticate to the cloud storage server Bob with account $(h(F), F)$, then Bob will reply with the ciphertext of the trapdoor τ_F. Alice can recover τ_F by decrypting the ciphertext with F as decryption key. With this short secret trapdoor τ_F, Alice may safely remove F from her local storage. Later, Alice can retrieve F back on demand from the cloud storage server using the trapdoor τ_F: Alice downloads a ciphertext of F from Bob after authenticating[1] herself with information $h(F)$ and τ_F, and decrypt the ciphertext to recover F with τ_F as decryption key.

Thus, client-side deduplication is completed by extracting a short secret trapdoor τ_F during the authentication with account $(h(F), F)$ to the cloud server Bob: Alice can obtain the correct secret trapdoor τ_F, if and only if she knows the decryption key which is just the file F.

1.1.3. Client-side Sharing. Current server-side sharing mechanism may not work with encrypted data, since the server itself has no access to the plaintext.

Our solution supports efficient client-side sharing mechanism: To share file F with Carol, Alice just passes the corresponding secret trapdoor τ_F and hash value $h(F)$ to Carol in a secure channel. With the trapdoor τ_F, Carol can convince Bob to accept w.r.t. $h(F)$, and thus is allowed to download the ciphertext of F from the cloud storage Bob. Then Carol can decrypt the ciphertext using τ_F as decryption key to recover

1. Indeed, Alice with $(h(F), \tau_F)$ can authenticate herself to Bob w.r.t. identity $h(F)$. That is the reason we call the short secret value τ_F as "trapdoor" of F.

file F. We emphasize that different files are encrypted under independent secret keys.

1.1.4. Secure Cloud Storage Service without Registration. The File Based Authentication enables a secure cloud storage service without pre-registration: The first uploading of a file F (in encrypted form) to the cloud will function as registration of the user account $(id = h(F), pin = F)$ to the cloud server automatically, and any owner of the same file F can authenticate to the server with $h(F)$ as id and F as password. The sensitive file F is the a priori shared secret among all owners of F, although these owners may have no knowledge of each other. Registration-free cloud service together with anonymous network routing [31], [13] make it much more difficult for the cloud server to track users' profile (like the user-file ownership graph) and usage history.

1.2. Organization

The rest of this paper is organized as below: We brief our main idea in the next Section 2 and introduce the background and related works in Section 3. We present the formulation of File Based Authentication in Section 4, and propose a construction and prove its security in Section 5. In Section 6, we apply the proposed File Based Authentication scheme to construct a secure cloud storage system which allows client side deduplication and sharing of encrypted data. At the end, Section 7 concludes this paper.

2. Overview

Our main idea is as below: Alice has file F in her local storage. She chooses two strings, denoted as k and `id`, randomly and independently. She encrypts the file F with the first random string k as encryption key using AES, to produce a ciphertext C. She somehow encrypts both strings k and `id` with file F as encryption key, using some special encryption method, to produce a short public value t. Alice uploads (C, t) to a cloud storage server, where this uploading functions as registration of account $(h(F), F)$ into the cloud storage server (Assume this is the first uploading of F to the cloud).

Another user Carol, who also has file F in local storage, can authenticate herself to the cloud storage server with account $(h(F), F)$ in this way: Carol sends $h(F)$ to the cloud storage server Bob, and Bob replies with t. Carol decrypts t with F as decryption key to recover the two secret strings k and `id`. Then Carol can convince Bob that she has the identity `id` through a Zero Knowledge Proof of Identity.

Later, Alice or Carol may authenticate to Bob with the two strings k and `id` and without F. She can download the ciphertext C of file F back from Bob by authenticating herself using information `id`, and then decrypt C with decryption key k, to retrieve file F back. Thus, the two short secret strings k and `id` serve as trapdoor for the long file F, where `id` allows Alice/Carol to download the ciphertext of F from Bob and k allows her to decrypt the ciphertext and recover F. This short trapdoor allows Alice and Carol to remove F from local storage (if they want to do so), and retrieve back F through Bob on demand.

In the above description, there is an unresolved issue: How to encrypt a short message with a long message F as encryption key, where F may have very low entropy compared to its bit-length? We employ an almost universal hash family to extract a short random value from F. Leftover hash lemma will guarantee that the extracted short random value is statistically close to true randomness. Then we encrypt the plaintext with the short random value as encryption key, using a one-time pad encryption scheme.

We remark that information leakages in the encryption component, authentication component and the randomness extraction component may have mutual inference. The security of overall system requires rigorous analysis, which is very challenging.

3. Background and Related Works

3.1. Proofs of Ownership

Halevi *et al.* [18] proposed Proofs of Ownership schemes to resolve the security flaw in the client-side deduplication mechanism implemented in several cloud storage services, including Dropbox. However, Proofs of Ownership schemes [18] has several limitations.

First, PoW schemes [18] cannot be applied to encrypted data and so cannot prevent the cloud storage server from accessing the user data stored in the cloud. Second, although PoW alone achieves strong security, a cloud storage server integrated with PoW schemes still

suffers from data-leakage-amplification. For example, the user account (i.e. user id and password) of (one of) the owner of a data file can be considered as a "short" representation of that file. Even more worse, the root account of the cloud storage system can be viewed as a "short" representation of all data files stored in the cloud storage.

3.2. Randomness Extraction

Randomness extraction [15], [24], [22], [8], [14] studies extract short almost-uniform randomness from a long input which has low min-entropy, or from noise data or from physical phenomenon and so on.

3.3. Universal Hash Family and Leftover Hash Lemma

In a ρ-Universal Hash Family [7], any two inputs is a collision with probability at most ρ where the probability is taken over hash function chosen uniformly from the hash family. It turns out such hash family can serve as a good randomness extractor, which extract a short almost-uniform randomness from a long input with low entropy. Leftover hash lemma [30], [4] quantifies the amount and quality of the extracted randomness. Performance of universal hash family is studied in [25].

Lemma 1 (Leftover Hash Lemma [30], [4])**.** *Assume that the family $\mathcal{H}$ of hash function $\mathcal{F}_s : \mathbb{F} \to \{0,1\}^\lambda$ is a $\frac{1+3\delta^2}{2^\lambda}$-universal hash family. If the min-entropy of F conditional on Z is at least $\lambda + 2\log(1/\delta)$, then*

$$\mathsf{SD}((h_s(F), s, Z),\ (U_\lambda, s, Z)) \leq \delta,$$

where SD *denotes the statistical difference.*

3.4. Zero Knowledge Proof of Identity

Zero Knowledge Proof of Identity [17] is a proof of knowledge of an identity which could be a uniformly random element from a space, without revealing any new information about this identity. We describe a variant version of Zero Knowledge Proof of Identity of Wu and Stinson [33] in Appendix A. Both this variant scheme (Lemma 4) and Feige and Fiat and Shamir (Theorem[2] 5 in [17]) scheme satisfy the below property:

Definition 1 (Zero Knowledge Proof of Identity)**.** *An ZKP of Identity scheme* $\mathbf{I} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *is (ϵ, δ)-secure, if for any PPT malicious prover $\mathcal{P}$ which can convince a verifier algorithm* Verify *that $\mathcal{P}$ has the identity* id *with probability at least ϵ, then $\mathcal{P}'$ indeed knows the identity* id *with probability at least $\epsilon - \delta$, where* id *is generated by the key generating algorithm* KeyGen.

4. Formulation

4.1. Generalized Semantic Secure Symmetric Encryption scheme

Before defining the security requirement for a File Based Authentication scheme, we generalize the notion of semantic-secure for symmetric encryption scheme. Let us define a Semantic-Secure game $\mathsf{Game}_{\epsilon,\zeta}$ for a symmetric encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ as below.

Commit. The adversary $\mathcal{A}$ chooses a set $\mathbb{M}$ of equal length bit-strings, and a distribution Π over the set $\mathbb{M}$, such that the min-entropy of distribution Π is at least $\log(1/\epsilon)$. The adversary $\mathcal{A}$ gives $(\mathbb{M}, \Pi)$ to the challenger $\mathcal{C}$.

Challenge. The challenger $\mathcal{C}$ samples a message M from the set $\mathbb{M}$ under distribution Π. Next, the challenger randomly and independently chooses ζ number of encryption keys $k_i := \mathsf{KeyGen}(1^\lambda), i \in [\zeta]$, and encrypts the message M under each key k_i to obtain a ciphertext $C_i := \mathsf{Enc}_{k_i}(M)$. All ζ ciphertexts $(C_1, \ldots, C_\zeta)$ are given to the adversary $\mathcal{A}$.

Guess. The adversary $\mathcal{A}$ outputs a guess $M' \in \mathbb{M}$, and wins this game if and only if $M' = M$.

Definition 2. *We say a symmetric encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is $(\epsilon, \delta, \zeta)$-Semantic-Secure under Cross-Key Ciphertext-Only Attack, if for any PPT adversary $\mathcal{A}$, the probability*

$$\mathsf{Pr}[\mathcal{A}\ \textit{wins}\ \mathsf{Game}_{\epsilon,\zeta}] \leq \epsilon + \delta.$$

2. Note that in [17], from factorization of RSA modulus n, one can find the secret identity with noticeable probability by solving square root and random guessing, since the identity size is only $O(\log\log n)$.

The standard semantic security can be considered as an extreme (strong) case: $(1/2,\ negl,\ 1)$-Semantic-Secure under Cross-Key Ciphertext-Only Attack.

Proxy re-encryption scheme might achieve such $(\epsilon, \delta, \zeta)$-semantic-security. Several negative results (e.g. [23]) on the security of RSA scheme under this attack model are known.

4.2. File Based Authentication: System Definition

A FBA scheme $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ consists of five algorithms $\mathcal{E}$, $\mathcal{D}$, $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{V}$.

- $\mathcal{E}(F, 1^\lambda) \to (\tau, C, t)$: The probabilistic encoding algorithm $\mathcal{E}$ takes as input a data file F and a security parameter λ, and outputs a secret trapdoor τ, a ciphertext C and a public token t.
- $\mathcal{D}(\tau, C) \to F$: The deterministic decoding algorithm takes as input a trapdoor τ and a ciphertext C, and outputs a file F.
- $\langle \mathcal{P}_1(F), \mathcal{V}(t) \rangle \to (\tau;\ \ b)$: The prover algorithm $\mathcal{P}_1$ with a file F as input interacts with the verifier algorithm $\mathcal{V}$ with a public token t as input. At the end of interaction, the prover algorithm $\mathcal{P}_1$ gets output τ and the verifier algorithm $\mathcal{V}$ gets output $b \in \{\texttt{Accept}, \texttt{Reject}\}$.
- $\langle \mathcal{P}_2(\tau), \mathcal{V}(t) \rangle \to (\perp;\ \ b)$: The prover algorithm $\mathcal{P}_2$ with trapdoor τ as input interacts with the verifier algorithm $\mathcal{V}$ with a public token t as input. At the end of interaction, the prover algorithm $\mathcal{P}_2$ gets no output and the verifier algorithm $\mathcal{V}$ gets output $b \in \{\texttt{Accept}, \texttt{Reject}\}$.

Definition 3 (Correctness)**.** *We say a* FBA *system* $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ *is* correct*, if the following conditions hold: For any data file* $F \in \mathbb{F}$ *and any positive integer* λ*, and* $(\tau, C, t) := \mathcal{E}(F, 1^\lambda)$*,*

- $\mathcal{D}(\tau, C) = F$.
- $\langle \mathcal{P}_1(F), \mathcal{V}(t) \rangle = (\tau,\ \texttt{Accept})$.
- $\langle \mathcal{P}_2(\tau), \mathcal{V}(t) \rangle = (\perp,\ \texttt{Accept})$.

4.3. File Based Authentication: Security Definition

Informally, in a FBA security game, the adversary first plays the role of verifier and talks to owners of file F to learn information, and then plays the role of prover, attempting to convince an honest verifier to accept the adversary as an owner of F, or try to find the value of F.

The FBA security game $\mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}(\lambda, \epsilon, \zeta)$ between a PPT adversary $\mathcal{A}$ and a challenger w.r.t. FBA scheme $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ is defined as below.

Setup. The challenger chooses a file F from domain $\mathbb{F}$, and runs the encoding algorithm to obtain $(\tau_0, C_0, t_0) := \mathcal{E}(F, 1^\lambda)$, where t_0 contains a hash value $h(F)$ of F as a part.

Leak. The adversary $\mathcal{A}$ has a one-time access to oracle $\mathcal{O}^F$. We emphasize that the oracle $\mathcal{O}^F$ is completely ignorant to (τ_0, C_0, t_0). After this oracle access, the challenger sends t_0 to the adversary $\mathcal{A}$. Until now, the file F has min-entropy at least $\log(1/\epsilon) \geq \lambda$ conditional on the adversary's view (This is our pre-condition).

Learning. The adversary $\mathcal{A}$ can adaptively make queries to the challenger, where each query is in one of the following forms:

- Setup query: The challenger responses the i-th setup query by running the probabilistic encoding algorithm on F to generate $(\tau_i, C_i, t_i) := \mathcal{E}(F, 1^\lambda)$ and sending (C_i, t_i) to the adversary. The adversary can make at most ζ queries in this type.
- Type-I Authentication query: The challenger, running the prover algorithm $\mathcal{P}_1$ with input F, interacts with adversary $\mathcal{A}$ which replaces the verifier algorithm $\mathcal{V}$, to obtain $b := \langle \mathcal{P}_1(F), \mathcal{A} \rangle$. The adversary knows the value of b, and can make polynomially many queries in this type.
- Type-II Authentication query (i): If $i > 0$ and the adversary $\mathcal{A}$ has not made i number of setup queries yet, the challenger ignores this query and does nothing. Otherwise, the challenger, running the prover algorithm $\mathcal{P}_2$ with input τ_i, interacts with adversary $\mathcal{A}$ which replaces the verifier algorithm $\mathcal{V}$, to obtain $b := \langle \mathcal{P}_2(\tau_i), \mathcal{A} \rangle$. The adversary knows the value of b, and can make polynomially many queries in this type.

Commit. The adversary $\mathcal{A}$ commits to a token t_{i^*} with $i^* \in [0, \zeta]$.

Guess. The adversary $\mathcal{A}$ outputs a guess F' and wins if $F' = F$.

Forge. The challenger, running the verifier algorithm $\mathcal{V}$ with input t_{i^*}, and interacts with the adversary $\mathcal{A}$, which replaces the prover algorithm $\mathcal{P}_2$, ($\mathcal{P}_1$, respectively), to obtain result $(\perp, b^*) := \langle \mathcal{A}, \mathcal{V}(t_{i^*}) \rangle$,

where $b^* \in \{\texttt{accept}, \texttt{reject}\}$. The adversary wins if $b^* = \texttt{accept}$.

We remark that we allow the adversary to make setup query in the above security game, to capture the possibility that a third party attacker impersonates the cloud storage Bob to lure Alice and Carol to upload their file F (e.g. phishing attack).

Definition 4. *We say a* FBA *system* $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ *is* $(\epsilon, \delta, \zeta)$*-Secure, if for any PPT adversary* $\mathcal{A}$*, the following conditions hold: Let* F' *and* b^* *be as in the* **Guess** *phase and* **Forge** *phase in game* $\mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}(\lambda, \epsilon, \zeta)$*,*

- $(\epsilon, \delta, \zeta)$*-privacy-preserving:* $\mathsf{Pr}[F' = F] \leq \delta$.
- $(\epsilon, \delta, \zeta)$*-unforgeable:* $\mathsf{Pr}[b^* = \texttt{accept}] \leq \delta$.

5. File Based Authentication Scheme

5.1. Overview

The idea is that: The sensitive data file is encrypted using an encryption scheme. We allow a client to access the private encryption key if and only if the client indeed has access to the exactly same file. Informally, we encrypt secret information using the data file as the private encryption key.

5.2. Construction

Let $\mathbf{I} = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ be a Zero Knowledge Proof of Identity scheme. and $\mathsf{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption scheme (e.g. AES). Let h be a full domain secure hash function (e.g SHA256) and $\{\mathcal{H}_s\}$ be a keyed-hash family. Let $\oplus$ denote the XOR operator between two equal length bit-strings. We construct a FBA scheme $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ as below.

5.2.1. Main Construction.

$\mathcal{E}(F, 1^\lambda)$**.**

1) $k_1 := \mathsf{E.KeyGen}(1^\lambda)$ and $k_2 := \mathsf{E.KeyGen}(1^\lambda)$, where $k_1, k_2 \in \{0,1\}^\lambda$
2) $(\texttt{id}, \texttt{PK}) := \mathbf{I}.\mathsf{KeyGen}(1^\lambda)$, where id is uniformly random over $\{0,1\}^{\gamma\lambda}$
3) $s_1, s_2 \xleftarrow{\$} \{0,1\}^\lambda$
4) $\tau = k_1 \| \texttt{id}$
5) $C := \mathsf{E.Enc}_{k_1}(F)$ and $C_{\texttt{id}} := \mathsf{E.Enc}_{k_2}(\texttt{id})$
6) $t := (s_1,\ s_2,\ \mathcal{H}_{s_1}(F) \oplus k_1,\ \mathcal{H}_{s_2}(F) \oplus k_2,\ C_{\texttt{id}}, \texttt{PK},\ h(F))$
7) Output (τ, C, t).

$\mathcal{D}(\tau, C)$**.**

1) Parse τ as $k_1 \| \texttt{id}$ and extract k_1.
2) $F' := \mathsf{E.Dec}_{k_1}(C)$
3) Output F'.

$\langle \mathcal{P}_1(F'),\ \mathcal{V}(t) \rangle$**.**

V1: Send t to the prover.

P1: Parse t as $(s_1,\ s_2,\ \mathcal{H}_{s_1}(F) \oplus k_1,\ \mathcal{H}_{s_2}(F) \oplus k_2,\ C_{\texttt{id}}, \texttt{PK},\ h(F))$. Compute the secret keys k_1', k_2' as below

$$k_1' := \mathcal{H}_{s_1}(F') \ \oplus\ (\mathcal{H}_{s_1}(F) \oplus k_1)$$
$$k_2' := \mathcal{H}_{s_2}(F') \ \oplus\ (\mathcal{H}_{s_2}(F) \oplus k_2)$$

Decrypt $C_{\texttt{id}}$ to obtain $\texttt{id}' := \mathsf{E.Dec}_{k_2'}(C_{\texttt{id}})$. The secret trapdoor $\tau' = k_1' \| \texttt{id}'$.

V$\leftrightarrow$P: Run the Zero Knowledge Proof of Identity protocol $\mathbf{I}$ between the prover and the verifier:

$$b := \langle \mathbf{I}.\mathsf{Prove}(\texttt{id}'),\ \mathbf{I}.\mathsf{Verify}(\texttt{PK}) \rangle$$

At the end of this procedure, the prover gets output τ' and the verifier gets output $b \in \{\texttt{accept}, \texttt{reject}\}$.

$\langle \mathcal{P}_2(\tau'),\ \mathcal{V}(t) \rangle$**.**

V1: Send t to the prover.

P1: Ignore verifier's first message t. Parse τ' as $k_1' \| \texttt{id}'$ and extract $\texttt{id}'$.

V$\leftrightarrow$P: Run the Zero Knowledge Proof of Identity protocol $\mathbf{I}$ between the prover and the verifier:

$$b := \langle \mathbf{I}.\mathsf{Prove}(\texttt{id}'),\ \mathbf{I}.\mathsf{Verify}(\texttt{PK}) \rangle$$

At the end of this procedure, the prover gets no output and the verifier gets output $b \in \{\texttt{accept}, \texttt{reject}\}$.

Notice that in the above construction, the (honest) verifier $\mathcal{V}$ cannot distinguish whether he/she is talking to prover $\mathcal{P}_1$ or $\mathcal{P}_2$.

Table 1: The differences between the alternative construction and the main construction in instruction level

Alternative Construction	Main Construction
Hash $\mathcal{H}_s : \mathbb{F} \to \{0,1\}^{\gamma\lambda}$	Hash $\mathcal{H}_s : \mathbb{F} \to \{0,1\}^{\lambda}$
$C_{\mathtt{id}} = \mathcal{H}_{s_2}(F) \oplus \mathtt{id}$	$C_{\mathtt{id}} = \mathsf{E.Enc}_{k_2}(\mathtt{id})$
No k_2	Has k_2
$t := (s_1,\ s_2,\ \mathcal{H}_{s_1}(F) \oplus k_1,$ $C_{\mathtt{id}}, \mathtt{PK},\ h(F))$	$t := (s_1,\ s_2,\ \mathcal{H}_{s_1}(F) \oplus k_1,$ $\mathcal{H}_{s_2}(F) \oplus k_2, C_{\mathtt{id}}, \mathtt{PK},\ h(F))$
All other differences implied by the above differences	All other differences implied by the above differences

5.2.2. Alternative Construction. The alternative construction is almost the same as the main construction in Section 5.2.1, except the following differences listed in Table 1.

Looking from the high level, the main construction requires that the file F has low min-entropy (at least λ) and can be proved in general case in the random oracle model; the alternative construction requires that the file F has high min-entropy (at least $\gamma\lambda$, where γ is determined by the underlying Zero Knowledge Proof of Identity protocol **I**), and can be proved in a simple case (when adversary does not make any set up queries, i.e. $\zeta = 0$, in the security game), in the standard model, by replacing random oracle with some almost universal hash family and Leftover Hash lemma.

5.3. Security Analysis

Theorem 2 (The Main Construction is Secure)**.** *Suppose hash functions h and $\mathcal{H}$ are random oracles, encryption scheme E is an $(\epsilon, \delta_1, \zeta+1)$-Semantic-Secure under Cross-Key Ciphertext-only Attack, and the Zero Knowledge Proof of Identity protocol* **I** *is (ϵ', δ_2)-secure with $\epsilon' > \delta_1 + \delta_2 + 2^{-\lambda}$. Then the above main construction $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ in Section 5.2.1 is a correct and $(\epsilon, \delta, \zeta)$-Secure FBA where $\epsilon \leq 2^{-\lambda}$, $\delta = \epsilon' > \delta_1 + \delta_2 + 2^{-\lambda}$ and integer $\zeta \geq 0$.*

Proof of Theorem 2: The correctness follows immediately from the correctness of the underlying encryption scheme and Zero Knowledge Proof of Identity scheme. We save the details. Now, we focus on the proof of security of the proposed scheme. We show the privacy preserving property first, and then prove the unforgeability property.

Privacy Preserving.

Claim 1. *There exists a PPT algorithm, which takes a ciphertext $C = \mathsf{E.Enc}_{k_1}(F)$ as input and outputs t, such that (C, t) is identically distributed as the last two outputs of $\mathcal{E}(F, 1^\lambda)$.*

We construct the conversion algorithm as below.

1) Input is $C = \mathsf{E.Enc}_{k_1}(F)$.
2) Choose key $k_2 := \mathsf{E.KeyGen}(1^\lambda)$ independently.
3) Choose $(\mathtt{id}, \mathtt{PK}) := \mathbf{I}.\mathsf{KeyGen}(1^\lambda)$ independently. Encrypt $\mathtt{id}$ to obtain $C_{\mathtt{id}} := \mathsf{E.Enc}_{k_2}(\mathtt{id})$.
4) Choose s_1, s_2, u_1, u_2, u_3 from $\{0,1\}^\lambda$ uniformly and independently.
5) Program the random oracle $\mathcal{H}$ by setting (conceptually) $\mathcal{H}_{s_1}(F) := u_1 \oplus k_1$ and $\mathcal{H}_{s_2}(F) := u_2 \oplus k_2$.
6) Program the random oracle h by setting $h(F) := u_3$.
7) Output is $(C, C_{\mathtt{id}},\ h(F),\ s_1,\ s_2,\ \mathcal{H}_{s_1}(F) \oplus k_1,\ \mathcal{H}_{s_2}(F) \oplus k_2)$, which is equal to $(C, C_{\mathtt{id}},\ u_3,\ s_1,\ s_2,\ u_1,\ u_2)$.

Until now, the proof of Claim 1 is complete. Note: In the above conversion, both k_1 and F are always unknown.

Claim 2. *Let F be a file which has at least $\log(1/\epsilon)$ min-entropy. Let $C_0, C_1, \ldots, C_\zeta$ be $(\zeta+1)$ ciphertexts, where each $C_i = \mathsf{E.Enc}_{k_i}(F)$ is a ciphertext of file F under key k_i and encryption keys $k_0, k_1, \ldots, k_\zeta$ are independently and randomly chosen. There exits a PPT algorithm $\mathcal{B}$, which takes $C_0, C_1, \ldots, C_\zeta$ as input, and can simulate the challenger in the security game $\mathsf{G}^{\mathsf{FBA}}_{\mathcal{A}}(\lambda, \epsilon, \zeta)$.*

Algorithm $\mathcal{B}$ runs as below:

1) For each ciphertext C_i, $i \in [0, \zeta]$, run the conversion algorithm specified in Claim 1 to obtain t_i. Furthermore, each identity $\mathtt{id}_i$ w.r.t. t_i is known.
2) In the **Leak** phase, give t_0 to the adversary.
3) Choose $u \xleftarrow{\$} \{0,1\}^\lambda$ at random, and program random oracle h by setting each $h(F)$ in t_i to the same value u.
4) For the i-th setup query made by the adversary, $i \in [\zeta]$, respond with (C_i, t_i).
5) For each Type-I or Type-II authentication query made by the adversary, follow the security game, with the secret information $\mathtt{id}_i$.

Notice that in $\mathsf{G}^{\mathsf{FBA}}_{\mathcal{A}}(\lambda, \epsilon, \zeta)$, the adversary can make at most ζ number of setup queries. Therefore, Claim 2 is

proved.

From Claim 2, we conclude that

$$\begin{aligned}&\mathsf{Pr}[\mathcal{A} \text{ finds } F \text{ in game } \mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}(\lambda,\epsilon,\zeta)]\\ \leq&\mathsf{Pr}[\mathcal{B}(C_0,\ldots,C_\zeta)=F]\\ \leq&\epsilon+\delta_1. \end{aligned} \quad (1)$$

The last inequality holds since the encryption scheme E is $(\epsilon,\delta_1,\zeta+1)$-Semantic-Secure.

Unforgeable. We just proved that for any PPT adversary $\mathcal{A}$, after interacting in the security game, the probability that $\mathcal{A}$ finds the file F is at most $\epsilon+\delta_1$. In a similar way, we can prove that, for any PPT adversary $\mathcal{A}$, after interacting in the security game, the probability that $\mathcal{A}$ finds the file $\mathtt{id}$ is at most $2^{-\lambda}+\delta_1$.

Recall that $\mathbf{I}$ is an (ϵ',δ_2)-secure Zero Knowledge Proof of Identity protocol with $\epsilon' > \delta_1+\delta_2+2^{-\lambda}$. Suppose the PPT adversary $\mathcal{A}$ can pass the authentication system with probability at least ϵ', then $\mathcal{A}$ knows the corresponding identity $\mathtt{id}$ with probability at least $\epsilon'-\delta_2 > \delta_1+2^{-\lambda}$. Contradiction! Therefore, no PPT adversary $\mathcal{A}$ can pass the authentication system with probability larger than or equal to ϵ'. That is,

$$\mathsf{Pr}[\mathcal{A} \text{ is accepted in game } \mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}(\lambda,\epsilon,\zeta)] \leq \epsilon'. \quad (2)$$

Now we combine results in privacy preserving aspect and the unforgeability aspect. Recall that, in privacy preserving aspect, we just showed that

$$\mathsf{Pr}[\mathcal{A} \text{ finds } F \text{ in game } \mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}(\lambda,\epsilon,\zeta)] \leq \epsilon+\delta_1. \quad (3)$$

Since $\epsilon \leq 2^{-\lambda}$ (it is a precondition), we have

$$\epsilon+\delta_1 \leq 2^{-\lambda}+\delta_1 \leq \epsilon'. \quad (4)$$

As a result, the proposed main construction is a $(\epsilon,\delta=\epsilon',\zeta)$-secure File based Authentication scheme.

□

Theorem 3 (The Alternative Construction is Secure when $\zeta=0$)**.** *Suppose $\{\mathcal{H}_s : \mathbb{F} \to \{0,1\}^{\gamma\lambda}\}$ is a $\frac{1+3\delta_1^2}{2^{\gamma\lambda}}$-universal hash family, and the Zero Knowledge Proof of Identity protocol* $\mathbf{I}$ *is (ϵ',δ_2)-Secure with $\epsilon' > 2^{-\gamma\lambda}+\delta_1+\delta_2$. Then the alternative construction in Section 5.2.2 is correct and (ϵ,δ,ζ)-Secure, where $\zeta=0$, $\log(1/\epsilon) > \lambda+2\log(1/\delta_1)$ and $\delta=\epsilon' > 2^{-\gamma\lambda}+\delta_1+\delta_2$.*

Notice that when $\zeta=0$, the adversary does not see any ciphertexts of the file F in the FBA security game, thus the mutual influence of information leakage in the encryption part and the randomness extraction part are avoided. In this simple case, we manage to prove the proposed scheme in standard model, using some almost universal hash function with Leftover Hash Lemma to replace the random oracle.

Proof of Theorem 3: The proof of correctness is straightforward. We focus on proof of security. After **Leak** phase of the security game, the adversary learns the hash function $h(F)$, and the file F has min-entropy larger than or equal to $\log(1/\epsilon) > \lambda+2\log(1/\delta_1)$, conditional on the view of the adversary. By the Leftover Hash Lemma 1, the statistical difference between the output of universal hash family $\{\mathcal{H}_s\}$ and true randomness over $\{0,1\}^{\gamma\lambda}$ is at most δ_1.

Since $\mathtt{id}$ is chosen uniformly randomly from $\{0,1\}^{\gamma\lambda}$, and protected in $\mathcal{H}_{s_2}(F)\oplus\mathtt{id}$ with $\mathcal{H}_{s_2}(F)$, which is δ_1-close to true randomness in $\{0,1\}^{\gamma\lambda}$, no (even if unbounded) adversary can guess the value $\mathtt{id}$ with probability larger than $2^{-\gamma\lambda}+\delta_1$. Since $\mathbf{I}$ is a (ϵ',δ_2)-secure Zero Knowledge Proof of Identity protocol, any PPT adversary $\mathcal{A}$ cannot convince the challenger to accept in the **Forge** phase of the security game with probability larger than $\epsilon' > 2^{-\gamma\lambda}+\delta_1+\delta_2$. Therefore,

$$\mathsf{Pr}[\mathcal{A} \text{ finds } F] \leq \mathsf{Pr}[\mathcal{A} \text{ is accepted in game } \mathsf{G}_{\mathcal{A}}^{\mathsf{FBA}}] \leq \epsilon'. \quad (5)$$

□

5.4. Open Problem

We realize that due to the mutual influence of information leakages in the encryption scheme and the universal hash, the (ϵ,δ,ζ)-security of the constructed FBA scheme (both main construction and alternative construction) with $\zeta \geq 1$ in the standard model, requires more rigorous analysis. We leave the construction of provable (ϵ,δ,ζ)-secure FBA with $\zeta \geq 1$ in the standard model as an open problem.

5.5. A Concrete Instantiation

5.5.1. Encryption Scheme. Our construction requires the underlying encryption scheme to be (ϵ,δ,ζ)-Semantic-Secure.

In real world application of our proposed scheme, we may choose AES-256 [9] (i.e. AES with 256 bits key) with random initial value as the underlying encryption scheme. To encrypt a file F under key k, choose a random initial value $\texttt{IV}$ independently and output ciphertext $(\texttt{IV}, \mathsf{AES}_k(\texttt{IV}, F))$. We emphasize that, AES encryption method has already been used widely, with implicitly assumed $(\epsilon, \delta, \zeta)$-Semantic-Security. For example, in SSL/TLS encrypted websites, a web page is encrypted using AES with different encryption key and different initial value per each request of that page.

It might be possible to construct provable $(\epsilon, \delta, \zeta)$-Semantic-Secure encryption scheme using proxy re-encryption scheme [6], [3].

5.5.2. Hash Function. We choose hash function SHA256 [26] as the full domain hash function h.

We choose almost universal hash family as the keyed-hash $\mathcal{H}$, for example, the AES-based universal hash family by Barak *et al.* [4].

We remark that some constructions of universal hash family (e.g. [12]) exploit property of linear system or modulo group/field. There might be a potential risk, that the original input F can be recovered from many number of hash outputs $\{(s, \mathsf{UHF}_s(F)\}$. In practice, we can set $\mathcal{H}_s(F) := (s,\ h(\mathsf{UHF}_s(F)))$.

5.5.3. Zero Knowledge Proof of Identity. One choice is the Feige and Fiat and Shamir [17] scheme, which requires $O(\lambda \log \lambda)$ bits long identity $\texttt{id}$, and $O(\lambda)$ round complexity.

An alternative choice is the ZKP of Identity scheme described in Appendix A, whose security is based Knowledge of Exponent Assumption [10]. In this scheme, the size of identity $\texttt{id}$ is $O(\lambda)$, and the round complexity is 1.

We remark that in both of above schemes, the identity is chosen from a group uniformly randomly. With proper coding scheme, we can choose identity uniformly randomly from a space $\{0,1\}^\lambda$ with proper λ.

6. Application of File Based Authentication in Cloud Storage Service

Let $(\mathcal{E}, \mathcal{D}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ be the File Based Authentication scheme, constructed in Section 5.2.1. In this section, we will apply this scheme to enable a registration-free secure cloud service, where each data file is encrypted on the client side, and client-side deduplication and sharing of encrypted files are supported. Furthermore, existing Proofs of Storage schemes (i.e Proofs of Retrievability [21], [28] and Provable Data Possession [2], [1]) and Proofs of Ownership schemes [18] can be incorporated with the proposed cloud storage service, in a seamless way.

We assume that the cloud storage server Bob is semi-honest, such that he will constantly maintain the integrity and availability of users' files, and yet he is curious to learn users' sensitive data. In this section, we only consider users' sensitive data file F which should be encrypted before uploaded to the cloud. In a real implementation of our proposed solution, the cloud server Bob should authenticate himself to each user before each interaction. For simplicity, the below description of our solution ignore the authentication of Bob's identity.

6.1. Cloud Storage Service without Registration

In a cloud storage service, many users store their files in a cloud storage server. These users may or may not keep a copy of their files in their local storage. They may download files from the server on demand, and share files among friends. To maintain a proper access control on users' files and prevent unauthorized access, each user is required to register an account in the cloud storage server before enjoying the cloud service.

A bit surprisingly, File Based Authentication can support a secure cloud storage service without pre-registration of user accounts. We can view our solution in this way: all owners of a file F share the same account $(h(F), F)$ with $h(F)$ as identity and F as password, although these owners may have few or no knowledge of each other. The first uploading of F into the cloud by some owner and the registration of account $(h(F), F)$ are in a single step. From the users' point of view, there is no additional registration process before using the cloud service, since the first use (w.r.t. a file F) of cloud service is considered as registration by the server. Compared with the typical cloud storage service with user registration [16], [32], our solution together with anonymous network routing [31], [13] can protect user privacy better, in the sense that it will

be more difficult for the cloud server to track users' profiles and usage history.

6.2. First Upload of file F

Suppose Alice is the first user who uploads data file F to the cloud storage server Bob. Alice chooses a security parameter λ and runs the encoding algorithm $\mathcal{E}$ to obtain $(\tau_F, C_F, t_F) := \mathcal{E}(F, 1^\lambda)$. Then Alice uploads the ciphertext C_F, the public token t_F and the hash value $h(F)$ to Bob, and keeps the secret trapdoor τ_F safely. Alice also keeps the hash value $h(F)$. Bob will keep tuple $(h(F), C_F, t_F)$ in the cloud storage. The above single process takes two functionalities simultaneously:

- Upload the ciphertext C_F of file F to the cloud storage server.
- Enroll account $(h(F), F)$ with user name $h(F)$ and password F to the cloud storage server.

6.3. Subsequent Upload of file F and Client-side Deduplication across users

Suppose Carol is one of the subsequent users who try to upload the file F' to Bob. Carol sends the hash value $h(F')$ to Bob. If Bob finds a matching tuple $(h(F'), C_{F'}, t_{F'})$ in his cloud storage, then a duplication event occurs. Next, Carol,who is running the prover algorithm $\mathcal{P}_1(F')$, interacts with Bob, who is running the verifier algorithm $\mathcal{V}(t_{F'})$. At the end of interaction, Carol obtains a secret trapdoor $\tau_{F'}$ and Bob obtains $b \in \{\texttt{Accept}, \texttt{Reject}\}$.

The correctness of our FBA scheme guarantees that if $F' = F$, then $\tau_{F'} = \tau_F$ and $b = \texttt{Accept}$. The security of our FBA scheme guarantees that if $F' \neq F$ and the min-entropy of file F conditional on Carol's view is at least ϵ (ϵ is specified in Theorem 2), then with overwhelming high probability that $\tau_{F'} \neq \tau_F$ and $b = \texttt{Reject}$.

We remark that even if (F', F) forms a collision of hash h, i.e. $h(F') = h(F)$ and $F' \neq F$, Carol still cannot get the trapdoor τ_F for F or convince honest Bob to accept, as long as the min-entropy of F conditional on Carol's view is at least ϵ.

6.4. Retrieving F from Cloud with short trapdoor

With the hash value $h(F)$ and the secret trapdoor τ_F at hand, Alice or Carol may remove F from her local storage (if she wants), and retrieve F back through the cloud storage server Bob using the trapdoor on demand.

In order to retrieve file F, Alice authenticates to Bob w.r.t. to user name $h(F)$: Alice, who is running prover algorithm $\mathcal{P}_2(\tau_F)$, interacts with Bob, who is running the verifier algorithm $\mathcal{V}(t_F)$, to get output $(\perp, b) := \langle \mathcal{P}_2(\tau_F),\ \mathcal{V}(t_F) \rangle$. Since the File Based Authentication scheme is correct, $b = \texttt{accept}$ with certainty. After successful authentication, Alice may request to download the ciphertext C_F from Bob, and then decrypt C_F using the trapdoor τ_F: $F := \mathcal{D}(\tau_F, C_F)$.

Carol may run the exactly same procedure as Alice does to recover file F from cloud storage server Bob.

6.5. Client-side Sharing

Current sharing mechanism in cloud storage is implemented on the server side: If a user Alice grants another user Carol to access Alice's file F, the cloud storage server Bob will provide a soft link of F to the user who can authenticate himself/herself with Carol's account id. With all files encrypted before uploaded to the cloud, the server itself is not able to access the content of users' files, and cannot grant any other users to access a particular file, neither. Therefore, the server-side sharing mechanism does not work with encrypted data. The cloud server can only grant users to download the encrypted data files with no means to decrypt them.

When all data files are encrypted, our solution allows client-side sharing between friends. To share a file F with Carol, Alice passes the corresponding secret trapdoor τ_F together with hash value $h(F)$ to Carol via a secure channel. With the secret trapdoor τ_F, Carol can convince the cloud storage server Bob to accept w.r.t. $h(F)$, and thus can download the ciphertext C_F of file F from Bob, where C_F can be decrypted using τ_F. We emphasize that in our scheme, different files will have independent encryption keys, even if they belong to the same owner.

6.6. Proofs of Storage and Proofs of Ownership

In this paper, we focus on the privacy protection of users' data files stored in the cloud storage against third party attackers, and the semi-honest cloud server, who will keep users' files intact but curious to know the sensitive information in users' file. If the data integrity is a concern, we may straightforwardly apply existing Proofs of Storage (PoS) schemes, including Proofs of Retrievability schemes [21], [28] and Provable Data Possession schemes [2], on the top of the solution proposed in this paper, taking the deterministic ciphertext C_F as input file for $\mathcal{POR}$ or $\mathcal{PDP}$ scheme. Additionally, the Proofs of Ownership (PoW) schemes [18] can also be applied directly on top of our scheme by taking the deterministic ciphertext C_F as input file.

For this purpose, we request the underlying encryption scheme of the File based Encryption scheme is deterministic. We emphasize that deterministic encryption with random initial value is allowed, as long as the initial value will be part of the resulting ciphertext.

Applications of PoS and PoW schemes upon file F is straightforward for the first user who uploads file F to the cloud storage server. We will focus on subsequent users who try to upload F to the cloud.

6.6.1. Proofs of Storage. Assume file F is in the cloud already. Carol, an owner of F, tries to upload F to the cloud. After interacting with the cloud storage server, Carol will know there is a copy of ciphertext C_F of file F in the cloud already. Carol authenticates herself to the cloud server with user name $h(F)$ and password F, and obtains the secret trapdoor τ_F. From the trapdoor $\tau_F = k_F \| \mathtt{id}_F$, Carol extracts the encryption key k_F and encrypts the file F in her local storage to obtain a ciphertext $C_F := \mathsf{E.Enc}_{k_F}(F)$. Then Carol runs the setup algorithm of an existing Proofs of Storage scheme, taking file C_F as input. Let $(C_F, \mathbf{R}, \mathbf{T}_F)$ denote the output, where $(C_F, \mathbf{R})$ is the error erasure encoded file of C_F and $\mathbf{T}_F$ is the authentication tag. Next, Carol sends both $\mathbf{R}$ and $\mathbf{T}_F$ to the cloud storage server Bob, and saves the communication bandwidth for the ciphertext C_F, since Bob has the same ciphertext already. This completes the setup phase of the PoS scheme w.r.t. file C_F between Carol and Bob. Later, Carol can verify integrity of file C_F stored in Bob's storage periodically, remotely and reliably, by running the authentication algorithm of the PoS scheme, without trusting in Bob and without downloading C_F.

We remark that, before removing F and C_F from local storage, Carol may run the authentication scheme of PoS scheme to ensure that Bob indeed has C_F.

6.6.2. Proofs of Ownership. Similar as above, Carol can obtain the secret trapdoor τ_F by authenticating to Bob with user name $h(F)$ and password F, and obtain ciphertext C_F by encrypting F in her local storage with key in the trapdoor τ_F. Suppose the PoW has been setup upon the ciphertext C_F by some other owner of F or by Bob. Then Carol can run the PoW scheme to convince Bob that she indeed owns C_F in local storage. Before deleting F and C_F from local storage, Carol may require Bob to prove that he indeed owns C_F using the PoW scheme. Such mutual proofs of ownership may prevent a faked cloud storage server (e.g. phishing attack) luring Carol to delete her files.

7. Conclusion

In this paper, we proposed a notion of "File Based Authentication", i.e. the owner of a file can authenticate himself/herself to a cloud server with the file as secret authentication key. We gave an efficient construction of File Based Authentication, and applied it to support secure client-side deduplication and sharing of encrypted data in cloud storage service. Furthermore, File Based Authentication implies that a secure cloud storage service without pre-registration of user accounts can be constructed. Building a secure cloud storage service over P2P [20], [27] network by integrating our solution with Proofs of Storage is in the future work. The possibility of registration-free and secure cloud computing for any other sorts of services rather than storage remains an open problem.

References

[1] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary Peterson, and Dawn Song. Remote data checking using provable data possession. *ACM Transactions on Information and System Security*, 14:12:1–12:34, 2011.

[2] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted

stores. In *CCS '07: ACM conference on Computer and communications security*, pages 598–609, 2007.

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9:1–30, 2006.

[4] Boaz Barak, Yevgeniy Dodis, Hugo Krawczyk, Olivier Pereira, Krzysztof Pietrzak, François-Xavier Standaert, and Yu Yu. Leftover Hash Lemma, Revisited. In *CRYPTO*, pages 1–20, 2011.

[5] Mihir Bellare and Adriana Palacio. The Knowledge-of-Exponent Assumptions and 3-Round Zero-Knowledge Protocols. In *CRYPTO '04: Annual International Cryptology Conference on Advances in Cryptology*, pages 273–289, 2004.

[6] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT '98*, page 127144, 1998.

[7] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions (extended abstract). In *Proceedings of the ninth annual ACM symposium on Theory of computing*, STOC '77, pages 106–112, 1977.

[8] Cline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sbastien Zimmer. Optimal randomness extraction from a die-hellman element. In *EUROCRYPT 2009*, page 572589, 2009.

[9] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. 2002.

[10] Ivan Damgård. Towards Practical Public Key Systems Secure Against Chosen Ciphertext Attacks. In *CRYPTO '91: Annual International Cryptology Conference on Advances in Cryptology*, pages 445–456, 1992.

[11] Alexander W. Dent. The Cramer-Shoup Encryption Scheme Is Plaintext Aware in the Standard Model. In *EUROCRYPT '06: Annual International Conference on Advances in Cryptology*, pages 289–307, 2006.

[12] Martin Dietzfelbinger, Joseph Gil, Yossi Matias, and Nicholas Pippenger. Polynomial Hash Functions Are Reliable (Extended Abstract). In *ICALP '92: International Colloquium on Automata, Languages and Programming*, pages 235–246, 1992.

[13] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation onion router. In *SSYM '04: Conference on USENIX Security Symposium*, pages 21–21, 2004.

[14] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes. In *CRYPTO '04*, pages 494–510, 2004.

[15] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[16] Dropbox. Dropbox. http://www.dropbox.com/.

[17] Uriel Feige, Amos Fiat, and Adi Shamir. Zero Knowledge Proofs of Identity. In *STOC '87: ACM symposium on Theory of computing*, pages 210–217, 1987.

[18] Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. In *CCS '11: ACM conference on Computer and communications security*, pages 491–500, 2011.

[19] Shulman-Peleg A. Harnik D., Pinkas B. Side Channels in Cloud Services: Deduplication in Cloud Storage. *IEEE Security and Privacy Magazine, special issue of Cloud Security*, 8(6), 2010.

[20] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas E. Anderson. Privacy-preserving P2P data sharing with OneSwarm. In *SIGCOMM*, pages 111–122, 2010.

[21] Ari Juels and Burton S. Kaliski, Jr. Pors: proofs of retrievability for large files. In *CCS '07: ACM conference on Computer and communications security*, pages 584–597, 2007.

[22] Hugo Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *CRYPTO '10*, pages 631–648, 2010.

[23] Alexander May and Maike Ritzenhofen. Solving systems of modular equations in one variable: how many RSA-encrypted messages does eve need to know? In *PKC '08*, pages 37–46, 2008.

[24] Michael Mitzenmacher and Salil Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *SODA '08: ACM-SIAM symposium on Discrete algorithms*, pages 746–755, 2008.

[25] Wim Nevelsteen and Bart Preneel. Software performance of universal hash functions. In *EUROCRYPT '99*, pages 24–41, 1999.

[26] NIST. National Institute of Standards and Technology. Secure hash standard (SHS). FIPS 180-2, August 2002.

[27] OneSwarm. OneSwarm. http://www.oneswarm.org/.

[28] Hovav Shacham and Brent Waters. Compact Proofs of Retrievability. In *ASIACRYPT*, pages 90–107, 2008.

[29] SpiderOak. SpiderOak. https://spideroak.com/.

[30] D. R. Stinson. Universal hash families and the leftover hash lemma, and applications to cryptography and computing. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 42:3–31, 2002.

[31] Paul Syverson, David Goldschlag, and Michael Reed. Anonymous Connections and Onion Routing. In *SP '97: IEEE Symposium on Security and Privacy*, pages 44–, 1997.

[32] Wikipedia. Comparison of online backup services. http://en.wikipedia.org/wiki/Comparison_of_online_backup_services.

[33] Jiang Wu and Doug Stinson. An Efficient Identification Protocol and the Knowledge-of-Exponent Assumption. Cryptology ePrint Archive, Report 2007/479, 2007. http://eprint.iacr.org/.

Appendix A. Zero Knowledge Proof of Identity

Here we give a simple Zero Knowledge Proof of Identity scheme based on Knowledge of Exponent Assumption [10], [11], [5], which can be considered as a variant of Wu and Stinson [33].

$\mathsf{KeyGen}(1^\lambda)$**.**

1) Find at random a group (p, g, G), where G is a multiplicative cyclic group of prime order p and g is a random generator of G.
2) $\mathtt{id} \xleftarrow{\$} \mathbb{Z}_p^*$.
3) $\mathtt{PK} := g^{\mathtt{id}}$.
4) Output $(\mathtt{id}, \mathtt{PK})$.

$\langle \mathsf{Prove}(\mathtt{id}),\ \mathsf{Verify}(\mathtt{PK}) \rangle$**.**

V1 : Choose $a, b \xleftarrow{\$} \mathbb{Z}_p^*$. Compute $h := g^b$. Send (h, h^a) to Prover.
P1 : Send $(A, B) = (h^{\mathtt{id}}, (h^a)^{\mathtt{id}},)$ to Verifier.
V1 : If $A^a = B$ and $A = \mathtt{PK}^b$, then accept; otherwise reject.

Assumption 1 (KEA [10], [11])**.** *For any PPT algorithm $\mathcal{A}$ which takes input (g, g^a) and random coin r and outputs (A, B) such that $A^a = B$ with probability at least μ, then there exists a PPT extractor algorithm $\bar{\mathcal{A}}$ which takes input (g, g^a, r) and outputs x such that $g^x = A$ with probability at least $\mu - \nu$, where ν is negligible in the security parameter.*

Lemma 4. *The above construction of proof of identity scheme is (μ, ν)-secure Zero Knowledge Proof of Identity, under Knowledge of Exponent Assumption 1.*

A.1. Proof of Knowledge of Identity

By Knowledge of Exponent Assumption, if there is a PPT algorithm which given (h, h^a) outputs (A, A^a), then there exists a PPT extractor which can find x such that $h^x = A$. Thus $g^{bx} = h^x = A = \mathtt{PK}^b = g^{\mathtt{id} \times b}$, which implies $bx = b \times \mathtt{id} \mod p$ and $x \mod p = \mathtt{id}$.

A.2. Zero Knowledge

Since the verifier himself/herself can compute prover's response $(A, B) = (\mathtt{PK}^b, \mathtt{PK}^{ab})$, the verifier gains no new knowledge from the interaction.