TrDup: enhancing secure data deduplication with user traceability in cloud computing

Jianfeng Wang

State Key Laboratory of Integrated Service Networks (ISN),
Xidian University,
Xi'an, P.R. China
Email: wjf01@163.com

Xiaofeng Chen*

State Key Laboratory of Integrated Service Networks (ISN),
Xidian University,
Xi'an, P.R. China

and

Fujian Provincial Key Laboratory of Network Security and Cryptology,
Fujian Normal University,
Fuzhou, P.R. China
Email: xfchen@xidian.edu.cn
*Corresponding author

Jin Li

School of Computer Science and Educational Software,
Guangzhou University,
Guangzhou, P.R. China
Email: lijin@gzhu.edu.cn

Kamil Kluczniak and Mirosław Kutyłowski

Department of Computer Science,
Wrocław University of Technology,
Wrocław, Poland
Email: kamil.kluczniak@pwr.edu.pl
Email: miroslaw.kutylowski@pwr.edu.pl

Abstract: Data deduplication is a special type of resource usage optimisation. It leads to reduction of the used storage space and network bandwidth by eliminating duplicate copies of the same data file. Convergent encryption, as the state-of-art approach, has been widely adopted to perform secure deduplication in the cross-user scenario. However, all prior solutions do not support user traceability: there is no way to trace the identities of malicious users in case of duplicate faking attacks. To cope with this problem, we propose a deduplication scheme called TrDup. It realises traceability of malicious user's identity by incorporating traceable signatures with message-locked encryption technique. The TrDup construction is followed by its formal security analysis.

Keywords: message-locked encryption; proof of ownership; secure deduplication; traceable signatures.

Reference to this paper should be made as follows: Wang, J., Chen X., Li, J., Kluczniak, K. and Kutyłowski, M. (2017) 'TrDup: enhancing secure data deduplication with user traceability in cloud computing', *Int. J. Web and Grid Services*, Vol. 13, No. 3, pp.270–289.

Biographical notes: Jianfeng Wang received MS in Mathematics and PhD in Cryptography from Xidian University, in 2013 and 2016, respectively. His research interests include applied cryptography and secure outsourced storage.

Xiaofeng Chen received BS and MS in Mathematics from Northwest University, China in 1998 and 2000, respectively. He got PhD in Cryptography from Xidian University in 2003. Currently he is a Professor at Xidian University. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. He is in the Editorial Board of IEEE Transactions on Dependable and Secure Computing (TDSC), Security and Communication Networks (SCN), Computing and Informatics (CAI), and International Journal of Embedded Systems (IJES) etc. He has served as the program/general chair or program committee member in over 30 international conferences.

Jin Li received BS (2002) in Mathematics from Southwest University and PhD in Information Security from Sun Yat-sen University in 2007. Currently he is a Professor at Guangzhou University. He has been selected as one of the science and technology new stars in Guangdong province. His research interests include security in cloud computing and applied cryptography. He has published over 80 research papers in refereed international conferences and journals and has served as the Program Chair or Program Committee Member in many international conferences.

Kamil Kluczniak was a PhD Student at Wrocław University of Technology during preparation of this work. He submitted a dissertation on domain signatures at Polish Academy of Sciences. He received MSc (2012) in Computer Science from Wrocław University of Technology. Most of the work presented in this paper has been done when during his visit at Xidian University in 2015. He participated in several research projects concerning privacy for digital identity documents. His research interests include provable security, pairing-based cryptography, privacy issues for authentication protocols, design of electronic signature schemes and domain signature schemes.

Mirosław Kutyłowski is a Full Professor at Wroclaw University of Technology. He is a member of Research Council of Institute of Computer Science at Polish Academy of Sciences and an elected member of Polish State Commission for Academic Titles. In his career, he was a Humboldt Fellow at Technical University of Darmstadt, Hochschuldozent at Heinz Nixdorf Institute at University of Paderborn, and professor at Institute of Computer Science, Wroclaw University. He has received MISTRZ Award from Foundation for Polish Science, IBM Faculty Award in Cyber Security and 2013 Award from Polish Chamber of Information Technology and Telecommunications, He has been active in different bodies concerning e-government issues, in particular concerning interface between ICT and legal systems. His research is focused on algorithms in distributed systems, privacy, security and cryptography.

This paper is a revised and expanded version of a paper entitled 'A new secure data deduplication approach supporting user traceability', presented at the *10th International Conference on Broadband and Wireless Computing, Communication and Applications*, Krakow, Poland, November, 2015, pp.120–124.

1 Introduction

Today we observe a rapid development of cloud computing. More and more individuals and enterprises move their own data into the cloud high-quality data services without maintaining local data systems. Moreover, the resource-constrained users can enjoy seemly unlimited computation resources by outsourcing computation-intensive task to the cloud server. Plenty of research works on outsourcing computation have been done (Atallah et al., 2002; Hohenberger and Lysyanskaya, 2005; Chen et al., 2014, 2015a,b).

Undoubtedly, current development of cloud computing can be attributed mainly to a strong commitment of the IT industry. Today's commercial cloud storage providers, such as Dropbox, Amazon S3 and Google Drive, provide online storage services, from simple backup services to cloud storage infrastructures (Harnik et al., 2010). As a result, an increasing amount of data are being outsourced into the cloud and the volume of such data increases almost exponentially. According to the recent analysis of IDC (Turner et al., 2014), the total volume of digital data that we create and copy annually will reach 44 ZB in 2020. Inevitably, this leads to a cost explosion of data storage. This concerns not only cost of the hardware and software necessary for keeping data but also rapidly growing energy consumption in storage systems. Therefore, one of the most critical challenges today is how to efficiently manage the ever-increasing datum and, at least, avoid wasteful resource utilisation. Deduplication - avoiding to store in the system the same data multiple times - is one of the important countermeasures against waste of storage space and has attracted considerable attention from both academic and industrial community.

Traditional data deduplication approach is as follows: after a given file is uploaded to the system for the first time by some user, all subsequent users do not have to perform upload operations, instead the system returns a link to the data copy already stored in the system. In case of data redundancy, this techniques enable us not only to save storage space but also significantly reduces communication overhead.

Unfortunately, data deduplication leads also to new security challenges. One of the most significant ones is that data deduplication is incompatible with traditional encryption. Specifically, to protect data confidentiality, users encrypt their files before outsourcing them to the cloud. If different encryption keys are used, then an identical data file shared by different users will result in different ciphertexts, which makes cross-user deduplication impossible. On the other hand, sharing the encryption keys between users might be practically impossible - users of the same data might be unaware of themselves. Moreover, personal data protection rules may prohibit showing who is holding a copy of a given file and therefore make distribution of the encryption keys nearly impossible.

As a promising solution, convergent encryption (CE) Douceur et al. (2002) encrypts/decrypts a data file with a *convergent* key derived from the cryptographic hash value of the file contents. Since CE uses a deterministic symmetric encryption scheme and the key depends only on the file contents, each copy generates the same ciphertext. This makes deduplication with encrypted data feasible. Bellare et al. (2013) defined a new

cryptographic primitive called message-locked encryption (MLE), which can be viewed as a generalisation of CE.

Furthermore, to enhance performance of deduplication, a randomised convergent encryption (RCE) scheme has been proposed. It can efficiently accomplish the necessary operations (i.e. key generation, message encryption and tag production). However, an Achilles heel of RCE is its vulnerability to so-called *duplicate faking attack*. Specifically, an honest user might be unable to retrieve his original file, since it can be replaced by a fake one in an undetectable way. To tackle this problem, an interactive version of RCE, called *interactive randomised convergent encryption* (IRCE), has been presented in Bellare and Keelveedhi (2015). In IRCE, a user can check consistency of a file tag by interacting with the server. In this way, the user can ensure that the original ciphertext is stored by the server. However, an adversary may upload a perverse ciphertext C' instead of the correct ciphertext C of a file $\mathcal{F}$. When the subsequent user intends to upload a ciphertext of $\mathcal{F}$, he gets a link to C'. He may interact with the system and find that C' is incorrect. However, the cloud server cannot check consistency between the tag and the ciphertext, since he has no access to the original plaintext. Thus, the cloud server cannot resolve which user is dishonest. From the point of view of practical applications, this is a major drawback. Preferably, it should be possible not only to identify which of these two users is malicious but also to trace him - i.e. identify all ciphertexts uploaded by him. This is nontrivial, if in principle a user might remain anonymous or appear under different identities.

1.1 Our contribution

In this paper, we focus on the problem of tracing malicious users performing duplicate faking attack against data deduplication systems. Our contribution is as follows:

- We introduce user traceability functionality in secure data deduplication. It enables to trace a malicious user in case of duplicate faking attack.
- We propose a concrete deduplication scheme TrDup enabling tracing of malicious users. Specifically, each user generates a kind of anonymous signature for the uploaded file - a variant of traceable signature scheme is used. Once a duplicate faking attack happens, the tracing agent can reveal the identity of the malicious user without revealing identities of other users or linking their files in the cloud system.
- We discuss security and efficiency issues of TrDup.

This is the full version of the paper that has been presented in BWCCA 2015 (Wang et al., 2015). The main differences between this paper and the conference version are as follows: First, we present the related work on secure data deduplication in Section 1.2. Second, We present the detailed security analysis of the proposed scheme in Section 5. Finally, we add a new Section 6 to provide a thorough experimental evaluation of the proposed scheme.

1.2 Related work

1.2.1 Secure deduplication

With the advent of the big data era, secure data deduplication has attracted considerable attention from the research community. Plenty of work on deduplication over encrypted data has been presented in the literature (Douceur et al., 2002; Storer et al., 2008; Abadi et al., 2013; Bellare et al., 2013; Keelveedhi et al., 2013; Stanek et al., 2014;

Li et al., 2014, 2015; González-Manzano and Orfila, 2015). Douceur et al. (2002) first introduced the idea of convergent encryption, which enables data confidentiality while performing deduplication. Bellare et al. (2013) formalised convergent encryption as MLE and explored its applications for creating space-efficient outsourced storage. Stanek et al. (2014) proposed a novel deduplication encryption scheme that can provide different security levels for data files according to their *popularity* that refers to how frequently the file is shared among users. In this way, they can achieve a more fine-grained trade-off between the storage efficiency and data security for the outsourced data. In order to improve confidentiality level for the outsourced data, Li et al. (2015) proposed a fine-grained deduplication mechanism based on user privileges. A user can perform a duplication check only for the files marked with matching privileges.

Unfortunately, all schemes mentioned above are vulnerable to the duplicate faking attack. As the first attempt, Bellare and Keelveedhi (2015) presented an IRCE scheme, which enables a user to check whether the correct ciphertext is stored by the cloud system. This procedure can be run by the user interacting with the cloud server during the phases of file upload and download. However, if an incorrect file is detected, we get no information about validity of other files stored in the system.

1.2.2 Group signatures

The system proposed in this paper reuses the idea of group signatures introduced by Chaum and van Heyst (1991). A group signature scheme allows any member of a group to anonymously sign a message on behalf of the group, while keeping the identity of the signer hidden. Only a group manager (or a group of users playing the role of a manager) can open the signature and identify the original signer in case of need. Group signature schemes have been widely adopted in many applications such as anonymous attestation (Brickell et al., 2004), identity escrow (Ateniese et al., 2000) and data integrity auditing (Wang et al., 2012).

Boneh et al. (2004) constructed a short group signature scheme based on the strong Diffie-Hellman assumption and the decision linear assumption in bilinear groups. According to this construction, the signatures have length 1,533 bits, which is below the recommended size of RSA signatures but provide a comparable security level. Nguyen and Safavi-Naini (2004) presented a group signature scheme based on bilinear parings. This scheme provides stronger anonymity features and be extended to support membership revocation.

So far, the existing group signature schemes enable tracing the signatures issued by a certain user by opening *all* signatures corresponding to a certain group. Note that this might be highly undesirable as privacy of the remaining users might be violated and we have to reveal much more data that we really wish. To address this issue, Kiayias et al. (2004) have introduced *traceable signatures*, where the group manager can generate a *tracing token* for each group user. With the tracing token, all signatures issued by a given user can be identified by a third party, called *Tracing Agent*, while neither revealing any information on the signer's identity nor linking the signatures of the other users. In a subsequent work (Choi et al., 2006), a short traceable signature scheme is constructed - its size is less than one-third of the size of the signature from Kiayias et al. (2004).

1.2.3 Proof of ownership

Halevi et al. (2011) introduced the concept of *proof of ownership* (PoW), which can be used to ensure data privacy and confidentiality in case of client-side deduplication.

Namely, a user can efficiently prove to the cloud storage server that he indeed owns a file without uploading it. Three concrete PoW constructions have been presented - they are based on a Merkle hash tree (MHT) built from the content of a data file. Specifically, a challenge/response protocol is performed between a server and a client. Each time the server requires the client to a valid verification object for the requested subset of MHT leaf nodes (the leaf nodes constitute the data file). Using a PoW, the cheat attack of malicious user can be prevented. That is, a user that knows only a hash signature of a file cannot convince the cloud server that he holds that file. Di Pietro and Sorniotti (2012) proposed an efficient PoW scheme, where each challenge is a seed for a pseudorandom generator and the response are the values in the file at bit positions derived by the generator from the seed. Every time a file is uploaded to the server, the latter computes a set of challenges for that file and stores them for a later check. Blasco et al. (2014) presented a PoW scheme based on a Bloom filter, which is efficient both on the server and the client side.

1.3 Paper organisation

The rest of the paper is organised as follows. In Section 2, we briefly present some preliminaries. In Section 3, we present the system and adversary model for the proposed deduplication scheme. A concrete deduplication scheme with user traceability is presented in Section 4. Section 5 is devoted to the security analysis of the proposed scheme. Its performance evaluation is given in Section 6. Finally, the conclusions are given in Section 7.

2 Preliminaries

2.1 Bilinear pairings

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic multiplicative groups of a prime order p, generated by g_1 and g_2, respectively. Let $\mathbb{G}_T$ be a cyclic multiplicative group of order p. A bilinear pairing is a mapping $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following properties:

1 bilinear: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$

2 non-degenerate: $e(g_1, g_2) \neq 1$

3 computable: there is an efficient algorithm to compute $e(u, v)$ for given $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$.

2.2 Complexity assumptions

Definition 2.1 (q-Strong Diffie-Hellman Assumption)**:** The q-Strong Diffie-Hellman (q-SDH) problem in ($\mathbb{G}_1$, $\mathbb{G}_2$) is defined as follows Boneh et al. (2004): given a tuple $(g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma^q)})$ of length $q + 2$ and $\gamma \in_R \mathbb{Z}_p^*$ as input, output a pair $(g_1^{1/\gamma+x}, x)$ where $x \in \mathbb{Z}_p^*$. We say that the q-SDH assumption holds in ($\mathbb{G}_1$, $\mathbb{G}_2$), if for every probability polynomial time algorithm $\mathcal{A}$, there exists a negligible function negl($\cdot$) such that

$$\Pr[\mathcal{A}(g_1, g_2, g_2^{\gamma}, \ldots, g_2^{(\gamma^q)}) = (g_1^{1/\gamma+x}, x) \text{ for some } x] \leq negl(\cdot).$$

Definition 2.2 (SDH Representation)**:** For $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ and $(u, v = g_2^{\gamma})$ with an unknown γ, an SDH representation is a tuple (A, x, t) with $A \in \mathbb{G}_1$ and $x, t \in \mathbb{Z}_p^*$ such that $A = (g_1^x \cdot u)^{1/\gamma+t}$.

Note that the tuple (A, x, t) satisfies $e(A, g_2^t \cdot v) = e(g_1^x \cdot u, g_2)$.

Definition 2.3 (Decision Linear Diffie-Hellman Assumption)**:** The decision linear Diffie-Hellman (DLDH) problem in $\mathbb{G}_1$ is defined as follows: Given $\{u, v, h, u^a, v^b, h^c\} \in \mathbb{G}_1^6$ as input, output *yes* if $a + b = c$ and *no* otherwise. We say that the DLDH assumption holds in $\mathbb{G}_1$ if for every probability polynomial time algorithm $\mathcal{A}$, there exists a negligible function negl$(\cdot)$ such that

$$\Big|\Pr[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = yes : u, v, h \xleftarrow{R} \mathbb{G}_1, a, b \xleftarrow{R} \mathbb{Z}_p] - \Pr[\mathcal{A}(u, v, h, u^a, v^b, \eta) = yes : u, v, h, \eta \xleftarrow{R} \mathbb{G}_1, a, b \xleftarrow{R} \mathbb{Z}_p]\Big| \leq negl(\cdot).$$

Definition 2.4 (Linear Encryption Boneh et al., 2004)**:** In linear encryption, a user's public key is a tuple of generators $u, v, h \in \mathbb{G}_1$, and the corresponding private key is $x, y \in \mathbb{Z}_p$ such that $u^x = v^y = h$. A ciphertext $LE(m)$ of $m \in \mathbb{G}_1$ is computed as follows:

$LE(m) := (u^a, v^b, m \cdot h^{a+b})$, where $a, b \xleftarrow{R} \mathbb{Z}_p$.

2.3 Bloom filter

A Bloom filter (BF) (Bloom, 1970) is a space-efficient data structure used to approximately represent a large set S and to perform membership queries over it, which consists of a binary array of size w, together with k independent hash functions $h_i : \{0, 1\}^\star \to [1, w]$ for $1 \leq i \leq k$. Initially, all positions of the array are set to 0. To insert an element x to the BF, we set the bit value to 1 at k positions depending on x, i.e. we set $BF[h_i(x)] = 1$ for $i = 1, 2, \ldots, k$.

A BF enables an efficient membership test: if there exists some $i \in \{1, 2 \ldots, k\}$ such that $B[h_i(x)] = 0$, then x is definitely not in the set S. Otherwise, we might assume that x is a member of S (which might be false with a probability growing with the number of elements inserted to the BF).

3 Problem formulation

3.1 System model

In this work, we consider a data deduplication system supporting user traceability, which involves four parties: the data user, the cloud server, the group manager and the tracing agency as illustrated by Figure 1.

Data User: A data user refers to an entity who wants to outsource data to the cloud server. In a data deduplication system, the user uploads a file only when a duplicate check shows that the file is still not stored in the cloud.

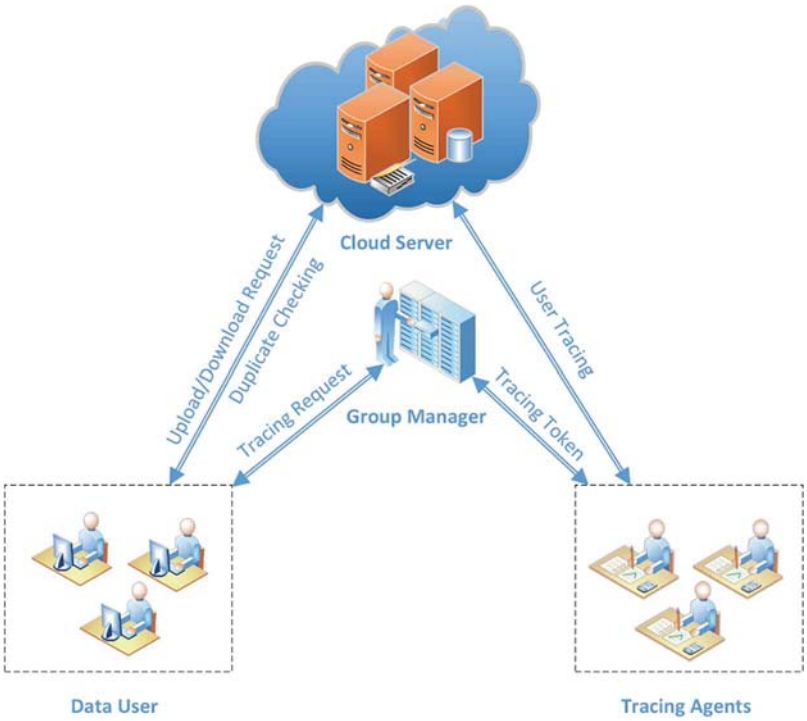
Cloud Server: The cloud server provides the data outsourcing service and stores data on behalf of the users. To reduce the storage cost, the cloud server stores only a one copy

of each file (of course, we mean here one *logical* copy, some physical redundancy might be necessary to resist hardware and software crashes).

Group Manager: The group manager is responsible for user enrollment and revocation. We assume that the group manager is trusted and does not collude with other entities. Besides, the group manager may reveal the identity of any malicious user and generates trace tokens for the tracing agency in order to trace all the signature generated by a malicious user.

Tracing Agent: The tracing agent is an entity that provides the user traceability service. Using the trace tokens from the group manager, the tracing agent may check whether data are uploaded by the malicious user. Note that the process of checking can be performed independently in parallel by many tracing agents.

Figure 1 Architecture for traceable secure deduplication (see online version for colours)



3.2 Adversary model

We assume that both the cloud server and the tracing agent are 'honest-but-curious'. That means, they follow the protocol, but try to find out as much secret information as possible based on data they hold and receive during the protocol execution. On the other hand, a malicious user would attempt to replace the original file ciphertext with a perverse one without being detected.

Based on the above assumptions, our adversary model considers two types of attackers:

1 An external attacker may obtain some knowledge (e.g. a hash value) of a data file. He plays the role of a data user and may interact with the cloud server. The goal of this adversary is to get access to the complete data file.

2 An internal attacker refers to a data user who tries to upload an incorrect ciphertext. The attacker may collude with some data users.

3.3 Design goals

Our main goal is to address the problem of user traceability in data deduplication system. According to the above attacker model, we aim to achieve the following design goals:

Data Soundness: We require that each user can perform deduplication operations and recover his original data file, even if the adversary exists and colludes with other malicious users. It implies that a malicious behaviour of an adversary will not violate the benefits of an honest user.

Data Confidentiality: We require that data aresecure against an adversary who does not own the data. That is, the user cannot get ownership of the data from the cloud server by running the PoW protocol if the user does not hold the file.

User Traceability: We require that the files uploaded by a certain user can be linked by a tracing agency in cooperation with the group manager. That is, in case of detection of misbehaviour, all files uploaded by a rogue user can be identified.

4 TrDup: a concrete construction

In this section, we present a concrete secure deduplication scheme that supports user traceability.

4.1 System description

- **Setup**(1^λ): Let 1^λ be the security parameter. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic multiplicative groups of a prime order p, let g_1, g_2 be generators of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively. Let e be a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let H, H_1 be two cryptographic hash functions, where $H : \{0,1\}^* \to \mathbb{Z}_p$, $H_1 : \{0,1\}^B \to \{0,1\}^l$, where B and l represent the block size and the token size, respectively. Let $P : \{0,1\}^l \times \{0,1\}^\star \to \{0,1\}^\kappa$ be a pseudorandom function. The Setup procedure is performed as follows:

  1 The group manager randomly selects $\gamma, \xi_1, \xi_2 \overset{R}{\leftarrow} \mathbb{Z}_p^*$, $m \overset{R}{\leftarrow} \mathbb{G}_1$, $h \overset{R}{\leftarrow} \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$, $\omega \overset{R}{\leftarrow} \mathbb{G}_2 \setminus \{1_{\mathbb{G}_2}\}$. Then computes $n = g_2^\gamma$, $u = h^{\xi_1^{-1}}$, $v = h^{\xi_2^{-1}}$. The system public key is published as $PK = \{m, n, \omega, u, v, h\}$. The master private key of the group manager is $MSK = \{\gamma, \xi_1, \xi_2\}$. Note that u, v, h are three generators of $\mathbb{G}_1$ and ω is a generator of $\mathbb{G}_2$.

2 In order to join the system, a user U_i randomly chooses $x_i \in \mathbb{Z}_p^*$ as his private key $usk_i = x_i$ and computes and sends $g_1^{x_i}$ to the group manager. Upon receiving a request, the group manager randomly selects $t_i \in \mathbb{Z}_p^*$ and computes $A_i = (g_1^{x_i} \cdot m)^{\frac{1}{\gamma + t_i}}$ and then sends (i, A_i, t_i) to the user U_i. The user U_i checks whether the equation $e(A_i, g_2^{t_i} \cdot n) = e(g_1^{x_i} \cdot m, g_2)$ holds and stores (i, A_i, t_i, x_i). The group manager adds a tuple $L_i = (g_1^{x_i}, A_i, t_i)$ to the user list L.

- **Encrypting a File**: Suppose that a user U_i wants to upload a data file $\mathcal{F}$. First, U_i picks at random a key K and computes a conventional ciphertext $C_1 = \text{Enc}(K, \mathcal{F})$ and a key $K_{\mathcal{F}} = \mathcal{H}(\mathcal{F})$. Here Enc is a symmetric encryption algorithm and $\mathcal{H}$ is a collision-resistant hash function. Then, the user U_i computes $C_2 = K \oplus K_{\mathcal{F}}$ and a tag $T_{\mathcal{F}} = \mathcal{H}(K_{\mathcal{F}})$ of the file $\mathcal{F}$. Furthermore, U_i initialises a Bloom filter $BF_{\mathcal{F}}$ and splits $\mathcal{F}$ into a set of blocks $\{B_i\}$ of the same length. For each block B_i, the user performs the following operations:

  1 U_i computes the block token $T_{B_i} = H_1(B_i)$ and a pseudorandom value $E_{B_i} = P(T_{B_i}, i)$;

  2 U_i inserts the E_{B_i} into the Bloom filter $BF_{\mathcal{F}}$, which will be used to prove the file ownership for the users.

  Finally, the user generates the final ciphertext $C_{\mathcal{F}} = (C_1, C_2, T_{\mathcal{F}}, BF_{\mathcal{F}})$.

- **Sign**: Before uploading the file ciphertext, the user U_i has to sign it with his private key in an anonymous manner. The details of the signing process using the private key (A_i, t_i, x_i) are described below:

  1 U_i selects $r_1, r_2, r_3 \overset{R}{\leftarrow} \mathbb{Z}_p$, at random, and computes $d_1 = t_i \cdot r_1$, $d_2 = t_i \cdot r_2$, and obtained the following values:
  $T_1 = u^{r_1}, \quad T_2 = v^{r_2}, \quad T_3 = A_i \cdot h^{r_1 + r_2}, \quad T_4 = \omega^{r_3}, \quad T_5 = e(g_1, T_4)^{x_i}$.

  2 U_i chooses $b_{r_1}, b_{r_2}, b_{d_1}, b_{d_2}, b_{t_i}, b_{x_i} \in \mathbb{Z}_p$ at random and then computes the following values:
  $B_1 = u^{b_{r_1}}, \quad B_2 = v^{b_{r_2}}, \quad B_3 = T_1^{b_{t_i}} \cdot u^{-b_{d_1}}, \quad B_4 = T_2^{b_{t_i}} \cdot v^{-b_{d_2}}$,
  $B_5 = e(g_1, T_4)^{b_{x_i}}$,
  $B_6 = e(T_3, g_2)^{b_{t_i}} \cdot e(h, g_2)^{-b_{d_1} - b_{d_2}} \cdot e(h, n)^{-b_{r_1} - b_{r_2}} \cdot e(g_1, g_2)^{-b_{x_i}}$.

  3 U_i computes a challenge $c = H(C_{\mathcal{F}}, T_1, \ldots, T_5, B_1, \ldots, B_6)$.

  4 With the aforementioned parameters, U_i computes the following values:
  $s_{r_1} = b_{r_1} + cr_1$, $s_{r_2} = b_{r_2} + cr_2$, $s_{d_1} = b_{d_1} + cd_1$, $s_{d_2} = b_{d_2} + cd_2$,
  $s_{x_i} = b_{x_i} + cx_i$, $s_{t_i} = b_{t_i} + ct_i$.

  5 The user U_i composes the signature σ for file $\mathcal{F}$ as

  $(T_1, \ldots, T_5, c, s_{r_1}, s_{r_2}, s_{d_1}, s_{d_2}, s_{x_i}, s_{t_i})$.

- **File Upload**: To upload a file $\mathcal{F}$, the user computes the tag $T_{\mathcal{F}}$ and sends it to the cloud server. Upon receiving this upload request, the cloud server first checks whether the tag $T_{\mathcal{F}}$ already exists.

1 If there is no duplicate on the cloud server, the user sends the ciphertext $C_{\mathcal{F}}$ as well as the signature σ to it. The cloud server will check the integrity of the uploaded file $\mathcal{F}$ as follows:

   a The cloud server computes the following values:
   $\tilde{B}_1 = u^{s_{r_1}} \cdot T_1^{-c}$, $\tilde{B}_2 = v^{s_{r_2}} \cdot T_2^{-c}$, $\tilde{B}_3 = T_1^{s_{t_i}} \cdot u^{-s_{d_1}}$,
   $\tilde{B}_4 = T_2^{s_{t_i}} \cdot v^{-s_{d_2}}$, $\tilde{B}_5 = e(g_1, T_4)^{s_{x_i}} \cdot T_5^{-c}$,
   $\tilde{B}_6 = e(T_3, g_2)^{s_{t_i}} \cdot e(h, g_2)^{-s_{d_1} - s_{d_2}} \cdot e(h, n)^{-s_{r_1} - s_{r_2}} \cdot e(g_1, g_2)^{-s_{x_i}} \cdot e(T_3, n)^{c} \cdot e(m, g_2)^{-c}$.

   b The cloud server checks correctness of the signature σ by checking the equation: $c \stackrel{?}{=} H(C_{\mathcal{F}}, T_1, \ldots, T_5, \tilde{B}_1, \ldots, \tilde{B}_6)$. If it holds, then the cloud server stores the file ciphertext $C_{\mathcal{F}}$ together with file signature σ and returns a link to these data to the user U_i.

2 If a duplicate of the tag $T_{\mathcal{F}}$ is found by the cloud server in the cloud storage, then the user has to perform a PoW protocol for the uploaded file by interacting with the cloud server. More specifically, the cloud server randomly chooses κ blocks, say $B_{k_1}, \ldots, B_{k_\kappa}$, and sends the set of their identities $T = \{k_1, \ldots, k_\kappa\}$ to the user. Upon receiving the set T, the user computes the tokens $T_j = H_1(B_{k_j})$, for $j \in [1, \kappa]$. Then the user sends the tokens T_j back to the cloud server. The cloud server performs the following operations:

   a For each $1 \leq j \leq \kappa$, the cloud server computes $E_{B_{k_j}} = P(T_j, k_j)$ with T_j received from the user.

   b The cloud server checks whether all E_{B_j} belongs to the $BF_{\mathcal{F}}$, if yes, a corresponding link is assigned to the user. Otherwise, the process is aborted.

   After the PoW check is accomplished, the cloud server computes $h = \mathcal{H}(C_1)$ and sends h, C_2 to the user. The user computes $K^{'} = C_2 \oplus K_{\mathcal{F}}$ and checks whether $\mathcal{H}(\text{Enc}(K^{'}, \mathcal{F})) \stackrel{?}{=} h$. If yes, then a link to the ciphertext of $\mathcal{F}$ will be assigned to the user. If the equality does not hold, then the user may claim that the ciphertext is invalid. In this case, he sends the file hash $\mathcal{H}(\mathcal{F})$ to the group manager and reports the problem.

- **User Trace**: Upon receiving the tracing request, the group manager first checks the correctness of the ciphertext to judge which user is dishonest. The procedure is as follows:

  1 The group manager computes the file tag $T^{'} = \mathcal{H}(\mathcal{H}(\mathcal{F}))$ and requests the corresponding ciphertext $C_{\mathcal{F}}$ from the cloud server.

  2 The group manager recovers the file encryption key $K^{'} = C_2 \oplus H(\mathcal{F})$ and obtains the file $\mathcal{F}^{'} = \text{Dec}(K^{'}, C_1)$. Finally, the group manager checks whether $H(\mathcal{F}^{'}) = H(\mathcal{F})$. If the equality holds, then the group manager claims that the ciphertext $C_{\mathcal{F}}$ is valid. Otherwise, the group manager will reveal the signer's identity by opening the signature with the following steps:

a The group manager computes $\tilde{A} = T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$ using the master private key MSK and then looks for an element A_k on the user list L such that which item satisfies the equation $A_k = \tilde{A}$.

b If the group manager has enough evidence that the user U_k is malicious, then he sends $g_1^{x_k}$ to all tracing agents.

c Using the token $g_1^{x_k}$, the tracing agent checks all file signatures. Namely, a given signature is linked to the malicious user if $e(g_1^{x_k}, T_4) = T_5$. All such signatures are returned to the group manager.

d The group manager deletes all links assigned to the malicious user.

Remark 1: Although the hash of the uploaded file needs to be revealed to the group manager, it will not cause security threat because the group manager is trusted by all users. As a complementary mechanism, if the group manager confirms that the ciphertext $C_{\mathcal{F}}$ is valid, it means that the user who launched the tracing request is dishonest, then he might be revoked from the system as a penalty. Otherwise the group manager will recover the identity index of the encipherer by opening the signature. Furthermore, a tracing token will be distributed to all tracing agents for tracing independently all the signatures generated by the dishonest user.

- **File Retrieve**: To download a file $\mathcal{F}$, the user sends the link for $\mathcal{F}$ to the cloud server and fetches the ciphertext $C_{\mathcal{F}}$ together with the signature σ. Finally, the user decrypts the ciphertext with his convergent key $K_{\mathcal{F}}$ to obtain the original plaintext $\mathcal{F}$.

4.2 Correctness of the proposed construction

We will show the correctness of our scheme according to the model defined in Choi et al. (2006).

- **Signature Correctness.**
  Note that

$$\tilde{B}_1 = u^{s_{r_1}} \cdot T_1^{-c} = u^{b_{r_1} + cr_1} \cdot (u^{r_1})^{-c} = B_1 \tag{1}$$

$$\tilde{B}_2 = v^{s_{r_2}} \cdot T_2^{-c} = v^{b_{r_2} + cr_2} \cdot (v^{r_2})^{-c} = B_2 \tag{2}$$

$$\begin{aligned} \tilde{B}_3 &= T_1^{s_{t_i}} \cdot u^{-s_{d_1}} = T_1^{b_{t_i} + ct_i} \cdot u^{-(b_{d_1} + cd_1)} \\ &= T_1^{b_{t_i}} \cdot (u^{r_1})^{ct_i} \cdot u^{-(b_{d_1} + ct_i \cdot r_1)} = B_3 \end{aligned} \tag{3}$$

$$\begin{aligned} \tilde{B}_4 &= T_2^{s_{t_i}} \cdot v^{-s_{d_2}} = T_2^{b_{t_i} + ct_i} \cdot v^{-(b_{d_2} + cd_2)} \\ &= T_2^{b_{t_i}} \cdot (v^{r_2})^{ct_i} \cdot v^{-(b_{d_2} + ct_i \cdot r_2)} = B_4 \end{aligned} \tag{4}$$

$$\begin{aligned} \tilde{B}_5 &= e(g_1, T_4)^{s_{x_i}} \cdot T_5^{-c} \\ &= e(g_1, T_4)^{b_{x_i} + cx_i} \cdot e(g_1, T_4)^{-cx_i} = B_5 \end{aligned} \tag{5}$$

$$\begin{aligned} \tilde{B}_6 = e(T_3, g_2)^{s_{t_i}} &\cdot e(h, g_2)^{-s_{d_1} - s_{d_2}} \cdot e(h, n)^{-s_{r_1} - s_{r_2}} \\ &\cdot e(g_1, g_2)^{-s_{x_i}} \cdot e(T_3, n)^{c} \cdot e(m, g_2)^{-c} \end{aligned}$$

$$
\begin{aligned}
&= e(T_3, g_2)^{b_{t_i}+ct_i} \cdot e(h, g_2)^{-(b_{d_1}+b_{d_2})-(cd_1+cd_2)} \\
&\quad \cdot e(h, n)^{-(b_{r_1}+b_{r_2})-(cr_1+cr_2)} \cdot e(g_1, g_2)^{-b_{x_i}-cx_i} \\
&\quad \cdot e(T_3, n)^c \cdot e(m, g_2)^{-c} \\
&= B_6 \cdot e(T_3, g_2)^{ct_i} \cdot e(h, g_2)^{-c(d_1+d_2)} \cdot e(h^{r_1+r_2}, n)^{-c} \\
&\quad \cdot e(g_1, g_2)^{-cx_i} \cdot e(T_3, n)^c \cdot e(m, g_2)^{-c} \\
&= B_6 \cdot e(T_3, g_2)^{ct_i} \cdot e(h^{r_1+r_2}, g_2)^{-ct_i} \cdot e(h^{r_1+r_2}, n)^{-c} \\
&\quad \cdot e(g_1, g_2)^{-cx_i} \cdot e(A_i \cdot h^{r_1+r_2}, n)^c \cdot e(m, g_2)^{-c} \\
&= B_6 \cdot e(A_i, g_2)^{ct_i} \cdot e(A_i, n)^c \cdot e(g_1, g_2)^{-cx_i} \cdot e(m, g_2)^{-c} \\
&= B_6 \cdot e(A_i, g_2^{t_i} \cdot n)^c \cdot e(g_1^{x_i} \cdot m, g_2)^{-c} = B_6
\end{aligned} \tag{6}
$$

Thus, $c = H(C_{\mathcal{F}}, T_1, \ldots, T_5, \tilde{B}_1, \ldots, \tilde{B}_6)$.

- **Open-Correctness.**
  For all signature generated by user u_i, we have

$$\tilde{A} = T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2}) = A_i \cdot h^{r_1+r_2}/[(u^{r_1})^{\xi_1} \cdot (v^{r_2})^{\xi_2}] = A_i.$$

- **Trace-Correctness.** Obviously, if U_k has created the signature containing T_4, T_5, then $e(g_1^{x_k}, T_4) = e(g_1, T_4)^{x_k} = T_5$ and the positive result is justified. On the other hand, for a signature created by other user U_i, we have $e(g_1^{x_k}, T_4) = e(g_1, T_4)^{x_k} \neq e(g_1, T_4)^{x_i} = T_5$.

5 Security analysis

Our scheme is designed to solve the user traceability problem in secure data deduplication in the case of a duplicate faking attack happens. Some basic tools have been adopted to construct our solution. In order to show the security of our construction, it is assumed that the underlying building blocks are secure. These basic tools include the RCE scheme, traceable signatures, and the proof of ownership PoW scheme. Based on this assumption, we show that TrDup is secure with respect to the following security issues.

5.1 Soundness of secure data deduplication

By soundness, we mean that if a user stores a file to the cloud system, then he should be able to recover the same file with a high probability or a party manipulating the file should be detected. This property should hold if the user actually uploads the file as well as in the case when the cloud system returns a link to a file already stored in the system.

Each file uploaded to the system is signed with an anonymous traceable signature, which covers the (double) hash $T_{\mathcal{F}}$ of the file. As shown in Choi et al. (2006), the traceable signatures are unforgeable based on the q-SDH assumption, hardness of discrete logarithm problem in $\mathbb{G}_1$, semantic security of linear encryption and properties of the hash function. This ensures that the uploaded ciphertext originates from a legitimate user. Of course, the first user uploading a file with a given tag may provide a perverse ciphertext $C_{\mathcal{F}}$. However, another user holding the file with the same tag can discover manipulations by obtaining *partial* ciphertext ($h = \mathcal{H}(C_1)$, C_2) from the cloud server after passing the PoW check. As he holds $K_{\mathcal{F}}$, the hash preimage of the tag, he can derive the encryption key $K = C_2 \oplus K_{\mathcal{F}}$

and then check whether $\mathcal{H}(\text{Enc}(K, \mathcal{F})) = h$. A successful manipulation would mean that the rogue user can create a file with the same hash value and therefore a collision for the hash function can be computed. In case when that the hash value does not match the decrypted file, the tracing functionality enables identification of the rogue user.

5.2 Confidentiality of the outsourced data

Confidentiality of files stored in the cloud with TrDup should be considered from two points of view. The first one is confidentiality against a party having access only to the ciphertext $C_{\mathcal{F}}$. Note that the data files are protected with the RCE scheme before outsourcing into the cloud server. According to the security notion of Bellare et al. (2013), RCE can achieve privacy against chosen distribution attack security, which guarantees that the encryptions of two unpredictable messages should be indistinguishable. That is, RCE can achieve semantic security when messages have high min-entropy.

The second important threat is that a rogue user U that has some *partial* information of a file $\mathcal{F}$ stored in the cloud gets access to the whole file. The necessary conditions are:

- U must know the tag $T_{\mathcal{F}}$ of the file $\mathcal{F}$
- execution of the PoW protocol must yield a positive result.

The second condition fails with a high probability if U holds only a fraction of blocks of file $\mathcal{F}$, the size of the Bloom filter ensures that the probability of a false positive is small and the number of trials κ during the PoW procedure is large enough.

Nevertheless, we should be aware that if U holds almost all blocks of $\mathcal{F}$, then the PoW protocol will give a positive result and U will get a link to the whole file. If U knows $K_{\mathcal{F}}$ as well, then he would be able to retrieve the whole file $\mathcal{F}$ and thereby learn the missing blocks.

5.3 Traceability of user identity

Based on the traceable signatures, the group manager can easily trace the identity of the signer for any signature. Specifically, the group manager computes $\tilde{A} = T_3/(T_1^{\xi_1} \cdot T_2^{\xi_2})$ using his master private key and then checks the user list to find $A_i = \tilde{A}$. Note that the group manager is trusted by all users and does reveal signer's identities.

On the other hand, the following attack scenarios may occur:

- **Framing Attack.** An adversary is allowed to act as a group manager and/or control some users. He attempts to construct a signature that traces to an innocent user or may claim a signature that was generated by an innocent user as its own. Note that the purpose of framing might be to prepare signatures under rogue data which can be used to cause revoking signatures created by an honest user via tracing functionality.
- **Anonymity Attacks.** The adversary is allowed to observe the operation of the system while the users are added, and they produce signatures. He may control some number of users as well as request tracing results for signatures of his choice. Then, he choose a message and two target users (not controlled by himself) and receives a signature of the message signed by one of these users - the one chosen at random by the challenger. Finally, the adversary has to guess which of the two signers produced the signature.

Fortunately, the detailed proofs show that the proposed traceable signature scheme resists the above attacks, which are already given in Choi et al. (2006).

6 Performance evaluation

In this section, we present results from a thorough experimental evaluation of TrDup. Our experiments are based on an implementation using Pairing-Based Cryptography (PBC) Library (http://crypto.stanford.edu/pbc) and OpenSSL Project (https://www.openssl.org) on a Linux machine with the Intel Core i5-3470 3.20 GHz CPU and 4 GB memory. We performed 100 runs for each test and take the average. Note that in order to precisely measure the overhead both at the cloud server and the user side, all experiments were performed on the same machine.

Table 1 presents the computation overhead of TrDup. Let P denotes a pairing operation, $\mathcal{E}xp$ denotes an exponentiation in $\mathbb{G}_1$, Exp denotes an exponentiation in $\mathbb{G}_T$, Enc denotes a symmetric encryption operation (resp., Dec stands for a decryption operation), $\mathcal{H}(\mathcal{F})$ denotes a hashing operation of file $\mathcal{F}$ (resp., by $\mathcal{H}(B)$ means a hashing operation of a file block B). Let n denote the number of users and $\mathcal{N}$ denote the number of signatures generated.

Table 1 Performance evaluation on TrDup

Procedure	*Pairings*	*Exponent. in* $\mathbb{G}_1$	*Exponent. in* $\mathbb{G}_T$	*Encryptions*	*Hashes*
Setup (server)	–	$(3+n)$	–	–	–
Setup (user)	2	2	–	–	–
Uploading a file	1	10	6	$1Enc$	$(\kappa+1)\mathcal{H}(B)$ $+2\mathcal{H}(\mathcal{F})$
Deduplication (cloud server)	3	8	7	–	$1\mathcal{H}(\mathcal{F})$
Deduplication (data user)	–	–	–	$1Enc$	$1\mathcal{H}(\mathcal{F})$
Tracing cost	$\mathcal{N}$	2	–	$1Dec$	$1\mathcal{H}(\mathcal{F})$

In the following experiments, we fix the number of hash functions of Bloom filter as $\kappa = 16$, the ratio of the length of Bloom filter and file block number is fixed to 23, then the false positive is approximately equal to 2^{-16}. We fix the block size as $B = 4$ KB, where each block will be inserted into the Bloom filter of the file. We implement cryptographic operations of hashing and encryption with the OpenSSL library. The operations of pairing and exponentiation are implemented with the PBC library.

In this paper, our experiments mainly focus on the computation overhead introduced by user tracing, including system setup, data outsourcing, data deduplication and user tracing. We evaluate the overhead by varying some specific factors, such as the number of system users, the file size and the number of signatures.

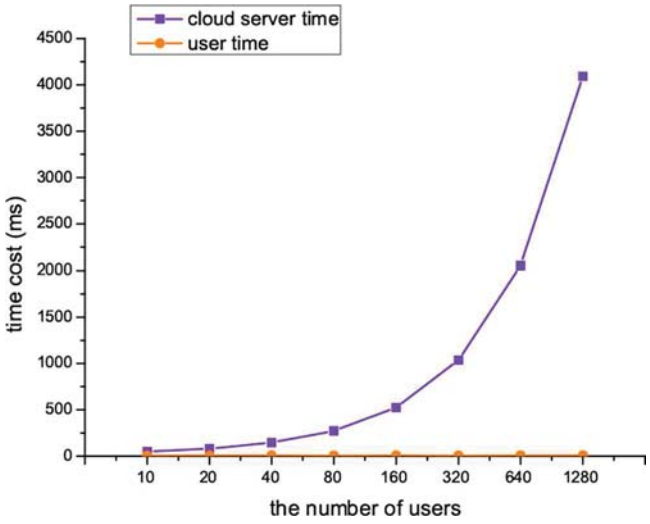
6.1 Computational complexity of system setup

In the system setup phase, the main computation overhead is dominated by pairing and exponentiation operations, which are used to generate system parameters (i.e. the pairs). More precisely, the cloud server needs to conduct $(3 + n)$ exponentiation operations, which is used to initialise the system parameters and generate anonymous identity tokens.

To join the TrDup system, each user conducts only two pairing and two exponentiation operations.

In our experiment, we have considered the number of users ranging from 10 to 1,280, at each step doubling the number of users. As shown in Figure 2, the cloud server needs about 4 s when the total number of users reach 1,000. Note that the cost of each user is constant. The result indicates that the time is acceptable in practice.

Figure 2 Impact of the number of system users on the system setup cost (see online version for colours)



6.2 Computational complexity of data outsourcing

In order to simultaneously achieve data privacy and deduplication, the data user could encrypt the data file with RCE before outsourcing it. The total computation overhead consists of three parts: RCE encryption, creating a traceable signature, and generation of a Bloom filter. Specifically, it includes 1 pairing, 10 exponentiations in $\mathbb{G}_1$, 6 exponentiations in $\mathbb{G}_T$, 1 RCE encryption, $(\kappa + 1)$ block hashings and 2 hash operations for creating $K_{\mathcal{F}}$ and $T_{\mathcal{F}}$. Note that the time cost of computing $K_{\mathcal{F}}$ is substantial and grows linearly with the file size. On Figure 3), it can be seen that the most computation time is stem from construction of the Bloom filter. Although the computation overhead is somewhat high, we remark that the outsourcing work is done once and most of the effort concerns offline activity. Still, the computation time is linear in the file size and linear on the parameter κ (note that the scale on Figure 3b is logarithmic!).

6.3 Computational complexity of data deduplication

Upon receiving an upload request, there are two cases:

- The cloud server claims that the file is uploaded for the first time and then checks the integrity of the uploaded ciphertext by verifying the signature.
- The file has been already uploaded by somebody else. The user retrieves part of ciphertext and checks the integrity of the ciphertext by recomputing file tag.

Figure 3 Impact of file size on data outsourcing time complexity. (a) Ciphertext generation time (b) Bloom filter generation time (see online version for colours)

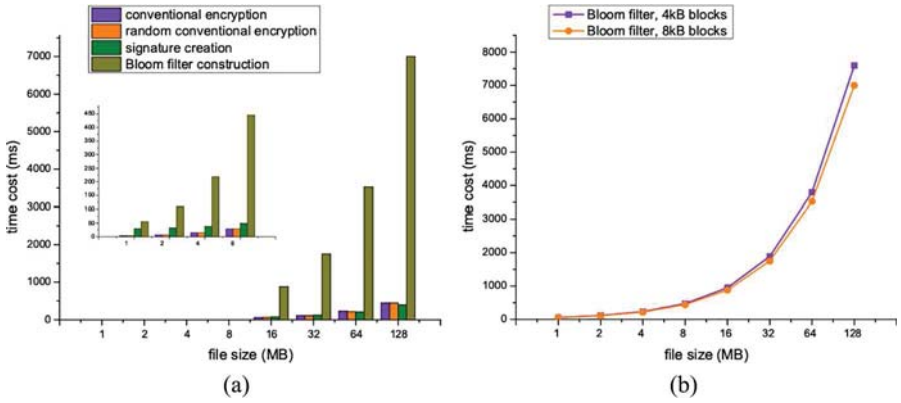
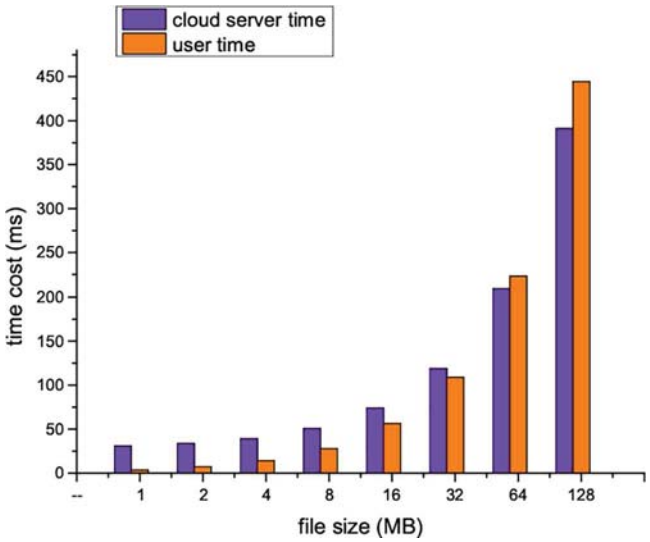


Figure 4 shows the time cost linearly increases with the file size. It should be pointed out that the cost of data user becomes higher than the cloud server's. The reason is that the cost of encryption and hash operation increases with the file size. On the other hand, the number of operations due to signature verification, such as pairing and exponentiation, is constant.

Figure 4 Impact of file size on data deduplication time complexity (see online version for colours)

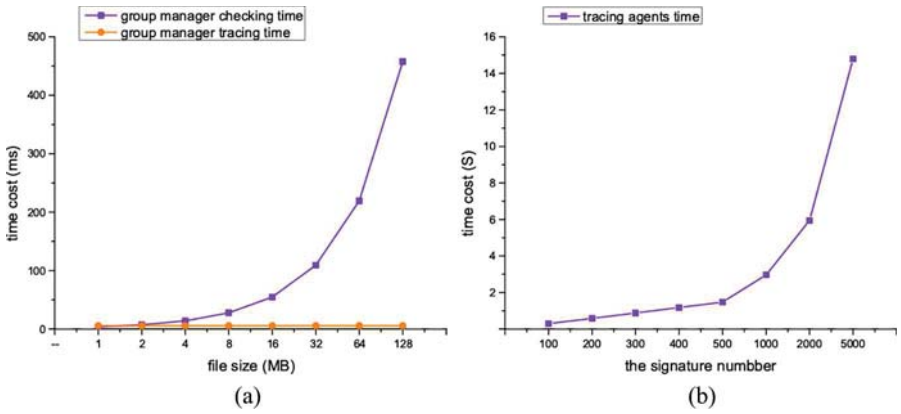


6.4 *Computational complexity of user tracing*

To trace the identity of the malicious user, the group manager first checks integrity of the ciphertext requested by the user. Then, the group manager generates the tracing token with his master key in a constant number of operations. Furthermore, the tracing token is distributed to all tracing agents to trace all the signatures generated by the malicious user.

Figure 5 shows that the user tracing is very efficient, e.g. it takes only 0.5 s to check a data file with the size of 128 MB. Although the computation overhead of the tracing agents is a little expensive when the number of signatures is large, all tracing agents can execute this operation in parallel. It shows that TrDup can be acceptable in a real-world scenario.

Figure 5 Impact of the file size and the number of signatures on user tracing time cost. (a) Group manager cost in user tracing (b) Tracing agent cost in user tracing (see online version for colours)



7 Conclusion

In this paper, we studied the problem of user traceability in secure data deduplication system. Note that the state-of-the-art secure deduplication scheme called IRCE does not consider the issue of identity tracing of the malicious user when a duplicate faking attack happens. To address the above issue, we introduced the concept of user traceability into secure data deduplication. Specifically, we have designed a novel concrete deduplication scheme called TrDup, which applies traceable signatures and PoW on random convergent encryption to achieve user traceability within data deduplication system. Security and efficiency evaluation show that TrDup can achieve the desired security goals, while providing a comparable computation overhead.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61572382), China 111 Project (No. B16037), National High Technology Research and Development Program (863 Program) of China (No. 2015AA016007), Doctoral Fund of Ministry of Education of China (No. 20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), the Fundamental Research Funds for the Central Universities (No. BDY151402) and Fujian Provincial Key Laboratory of Network Security and Cryptology Research Fund (Fujian Normal University) (No. 15009). This work was supported by Faculty of Fundamental Problems of Technology, Wrocław University of Technology. This work is also a result of Polish-Chinese cooperation

venture of Xidian University and Wrocławaw University of Technology on Secure Data Outsourcing in Cloud Computing.

References

Abadi, M., Boneh, D., Mironov, I., Raghunathan, A. and Segev, G. (2013) 'Message-locked encryption for lock-dependent messages', *Proceedings of Advances in Cryptology (CRYPTO'13)*, 18–22 August, 2013, Santa Barbara, CA, USA, pp.374–391.

Atalla, M.J., Pantazopoulos, K., Rice, J.R. and Spafford, E.E. (2002) 'Secure outsourcing of scientific computations', *Advances in Computers*, Vol. 54, pp.215–272.

Ateniese, G., Camenisch, J., Joye, M. and Tsudik, G. (2000) 'A practical and provably secure coalition-resistant group signature scheme', *Proceedings of Advances in Cryptology (CRYPTO'00)*, 20–24 August, 2000, Santa Barbara, California, USA, pp.255–270.

Bellare, M. and Keelveedhi, S. (2015) 'Interactive message-locked encryption and secure deduplication', *Proceedings of the 18th International Conference on Practice and Theory in Public-Key Cryptography (PKC'15)*, 30 March–1 April, 2015, Gaithersburg, MD, USA, pp.516–538.

Bellare, M., Keelveedhi, S. and Ristenpart, T. (2013) 'Message-locked encryption and secure deduplication', *Proceedings of EUROCRYPT 2013*, 26–30 May, 2013, Athens, Greece, pp.296–312.

Blasco, J., Di Pietro, R., Orfila, A. and Sorniotti, A. (2014) 'A tunable proof of ownership scheme for deduplication using Bloom filters', *Proceedings of 2014 IEEE Conference on Communications and Network Security (CNS'14)*, 29–31 October, 2014, San Francisco, CA, USA, pp.481–489.

Bloom, B.H. (1970) 'Space/time trade-offs in hash coding with allowable errors', *Communications of the ACM*, Vol. 13, No. 7, pp.422–426.

Boneh, D., Boyen, X. and Shacham, H. (2004) 'Short group signatures', *Proceedings of Advances in Cryptology (CRYPTO'04*, 15–19 August, 2004, Santa Barbara, California, USA, pp.41–55.

Brickell, E., Camenisch, J. and Chen, L. (2004) 'Direct anonymous attestation', *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS'04)*, 25–29 October, 2004, Washington, DC, USA, pp.132–145.

Chaum, D. and van Heyst, E. (1991) 'Group signatures', *Proceedings of EUROCRYPT 1991*, 8–11 April, 1991, Brighton, UK, pp.257–265.

Chen, X., Huang, X., Li, J., Ma, J. and Lou, W. (2015a) 'New algorithms for secure outsourcing of large-scale systems of linear equations', *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 1, pp.69–78.

Chen, X., Li, J., Ma, J., Tang, Q. and Lou, W. (2014) 'New algorithms for secure outsourcing of modular exponentiations', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 9, pp.2386–2396.

Chen, X., Susilo, W., Li, J., Wong, D.S., Ma, J., Tang, S. and Tang, Q. (2015b) 'Efficient algorithms for secure outsourcing of bilinear pairings', *Theoretical Computer Science*, Vol. 562, pp.112–121.

Choi, S., Park, K. and Yung, M. (2006) 'Short traceable signatures based on bilinear pairings', *Proceedings of the 1st International Workshop on Security (IWSEC'06)*, 23–24 October, 2006, Kyoto, Japan, pp.88–103.

Di Pietro, R. and Sorniotti, A. (2012) 'Boosting efficiency and security in proof of ownership for deduplication', *Proceedings of the 7th ACM Symposium on Information, Compuer and Communications Security (CCS'12)*, 2–4 May, 2012, Seoul, Korea, pp.81–82.

Douceur, J.R., Adya, A., Bolosky, W.J., Simon, D. and Theimer, M. (2002) 'Reclaiming space from duplicate files in a serverless distributed file system', *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2–5 July, 2002, Vienna, Austria, pp.617–624.

González-Manzano, L. and Orfila, A. (2015) 'An efficient confidentiality-preserving proof of ownership for deduplication', *Journal of Network and Computer Applications*, Vol. 50, pp.49–59.

Halevi, S. Harnik, D. Pinkas, B. and Shulman-Peleg, A. (2011) 'Proofs of ownership in remote storage systems', *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*, 17–21 October, 2011, Illinois, USA, pp.491–500.

Harnik, D., Pinkas, B. and Shulman-Peleg, A. (2010) 'Side channels in cloud services: deduplication in cloud storage', *IEEE Security & Privacy*, Vol. 8, No. 6, pp.40–47.

Hohenberger, S. and Lysyanskaya, A. (2005) 'How to securely outsource cryptographic computations', *Proceedings of the 2nd Theory of Cryptography Conference (TCC'05)*, 10–12 February, 2005, Cambridge, MA, USA, pp.264–282.

Keelveedhi, S., Bellare, M. and Ristenpart, T. (2013) 'DupLESS: server-aided encryption for deduplicated storage', *Proceedings of the 22th USENIX Security Symposium*, 14–16 August, 2013, Washington, DC, USA, pp.179–194.

Kiayias, A., Tsiounis, Y. and Yung, M. (2004) 'Traceable signatures', *Proceedings of EUROCRYPT 2004*, 2–6 May, 2004, Interlaken, Switzerland, pp.571–589.

Li, J., Chen, X., Li, M., Li, J., Lee, P.C. and Lou, W. (2014) 'Secure deduplication with efficient and reliable convergent key management', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 6, pp.1615–1625.

Li, J., Li, Y., Chen, X., Lee, P. C. and Lou, W. (2015) 'A hybrid cloud approach for secure authorized deduplication', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 26, No. 5, pp.1206–1216.

Nguyen, L. and Safavi-Naini, R. (2004) 'Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings', *Proceedings of ASIACRYPT 2004*, 5–9 December, 2004 Jeju Island, Korea, pp.372–386.

OpenSSL Project, Available at: https://www.openssl.org. (Accessed 10 July 2015).

Pairing-Based Cryptography Library, Available at: http://crypto.stanford.edu/pbc/. (Accessed 21 June 2015).

Stanek, J., Sorniotti, A., Androulaki, E. and Kencl, L. (2014) 'A secure data deduplication scheme for cloud storage', *Proceedings of the 18th International Conference on Financial Cryptography and Data Security (FC'14)*, 3–7 March, 2014, Christ Church, Barbados, pp.99–118.

Storer, M.W., Greenan, K.M., Long, D.D. and Miller, E.L. (2008) 'Secure data deduplication', *Proceedings of the 2008 ACM Workshop on Storage Security and Survivability (StorageSS'08)*, 31 October, 2008, Alexandria, VA, USA, pp.1–10.

Turner, V., Gantz, J.F., Reinsel, D. and Minton, S. (2014) 'The digital universe of opportunities: rich data and the increasing value of the internet of things', *International Data Corporation, White Paper*, IDC_1672, 2014.

Wang, B., Li, B. and Li, H. (2012) 'Knox: privacy-preserving auditing for shared data with large groups in the cloud', *Proceedings of the 10th International Conference on Applied Cryptography and Network Security (ACNS'12)*, 26–29 June, 2012, Singapore, pp.507–525.

Wang, J., Chen, X., Li, J., Kluczniak, K. and Kutyłowski, M. (2015) 'A new secure data deduplication approach supporting user traceability', *Proceedings of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA'15)*, 4–6 November, 2015, Krakow, Poland, pp.120–124.