2016 IEEE 8th International Conference on Cloud Computing Technology and Science

A Hybrid Deduplication for Secure and Efficient Data Outsourcing in Fog Computing

Dongyoung Koo
Department of Computer Science and Engineering
Korea University
Seoul, South Korea
Email: dykoo@korea.ac.kr

Youngjoo Shin
National Security Research Institute
Daejeon, South Korea
Email: yjshin@nsr.re.kr

Joobeom Yun
Department of Computer and Information Security
Sejong University
Seoul, South Korea
Email: jbyun@sejong.ac.kr

Junbeom Hur
Department of Computer Science and Engineering
Korea University
Seoul, South Korea
Email: jbhur@korea.ac.kr

***Abstract*—With prevalence of remote storage services, data privacy issues become more serious owing to loss of control to outsourced data. In the meanwhile, the service providers tend to minimize storage utility costs. To minimize the storage costs while preserving data privacy, secure deduplication techniques have been proposed, which are categorized into client-side or server-side approaches. Client-side approach achieves storage and bandwidth savings at the same time but allows external adversaries to know existence of duplicates in the remote storage. On the contrary, server-side one prevents the adversaries from getting acknowledged but sacrifices network bandwidth savings. In fog computing, however, which is a new computing paradigm extending the cloud computing by outsourcing a centralized workload of the cloud to geographically distributed fog devices located at the edge of the networks, the previous deduplication schemes cannot guarantee efficiency improvement and privacy preservation simultaneously. In this paper, we present a simple but nontrivial solution of these contradictory issues in fog storage. The proposed hybrid secure deduplication protocol combines client- and server-side deduplications by taking untrustworthy fog storage environments into account. The client-side deduplication is applied in inter-network (*i.e.*, cloud-fog network) communications to prevent network congestion at the network core, while the server-side deduplication is adopted in intra-network (*i.e.*, user-fog network) communications to prevent information leakage via side channels for maximal data privacy. Performance and security analyses demonstrate the comparable efficiency of the proposed scheme with security enhancement.**

***Keywords*-Data outsourcing, client-side deduplication, server-side deduplication, fog computing, data privacy, efficiency**

I. Introduction

Fog computing, as an extension of the cloud computing from the core to the edge of the network, is a promising future generation paradigm. Due to the rise of IoT devices with limited computing resources, cloud-based solutions have been extensively researched. However, forecasts based on the recent growth of the IoT market [1], [2] indicate that centralized clouds will be unlikely to be able to provide satisfactory services to end users in the near future. On the other hand, fog computing efficiently handles concentrated service requests by outsourcing the centralized workloads in terms of storage space and network bandwidth to fog devices, which are distributed over a wide geographical range [3], [4], [5]. While the central cloud provides the overall computing services, it also manages decentralized heterogeneous fog devices such as set-top box, access point, and home gateway. Individual fog devices located near IoT devices provide faster services to end users based on their own computation, storage, and network capabilities. In short, fog computing has the following attractive attributes: (1) low latency, (2) enhanced user experience (*i.e.*, high quality service), and (3) context awareness based on locational proximity to end users [6], [7].

The centralized cloud storage is unable to handle enormous volumes of data in a timely manner given a finite network bandwidth. Distributed storage, especially fog devices, is incapable of providing computing services to users owing to its limited resources and field of vision. Therefore, efficient resource management can be seen as one of the most important goals of commercial cloud storage services.

As regards space utilization, deduplication is an attractive data compression technique which stores only a single copy of duplicate data and provide owners with a link to it. Compared to cloud storage, fog devices located at the user side with temporal storage (owing to limited storage capacity) can perform deduplication and provide data outsourcing services to data owners faster than the ones in central cloud architecture. At the same time, the central cloud can efficiently utilize storage space from a global perspective by receiving and maintaining unique data from fog devices.

As regards bandwidth utilization, fog computing can be seen as three tier (cloud-fog-end user) layered network. The service delays are more likely to happen in communications between the central cloud and the distributed fogs, which is (possibly, multi-hop) *inter-network* communications. Contrariwise, *intra-network* (generally, single hop or a few ones) communications between the fog and end users have relatively low latency. Thus, client-side deduplication, which allows end users to upload only a small and unique instance

9781-5090-1445-3/16$31.00 © 2016 IEEE (CloudCom'16)

2330-2186/16 $31.00 © 2016 IEEE
DOI 10.1109/CloudCom.2016.51

IEEE computer society

of data, is adequate on the inter-network due to its bandwidth efficiency by preventing repetitive uploads of the whole duplicate data.

Despite the compelling benefits of deduplication, privacy issues surrounding outsourced data have also received close attention. Data owners cannot guarantee the secure management because they lose control on their outsourced data in remote storage systems after outsourcing [8], [9]. Thus, end users attempt to outsource encryptions of their contents while supporting deduplication such as convergent encryption (CE) [10]. In this context, server-side deduplication, which allows repetitive uploads of duplicate data but eliminates them at the server side, is more appropriate to communications between the fog and end users (on the intra-network) since it prohibits illegitimate users from learning side information such that duplicate content resides in remote storage or not.

Considering secure deduplication together with storage and bandwidth efficiency in the fog storage systems, we present a hybrid secure deduplication protocol. By applying client-side deduplication at inter-network level and server-side counterpart at intra-network level, the proposed scheme achieves best-effort bandwidth with desirable security guarantees. Specifically, our protocol satisfies the following properties:

1) End users can upload their contents without prior key agreement while preventing side information leakage.
2) Fog storage and cloud storage cannot learn any information about outsourced data except occurrences of deduplication.
3) Throughput increases by adaptively adopting client- and server-side deduplications according to network condition.
4) Storage utilization ratio increases in fog storage and cloud storage by exploiting multi-tier hierarchy in fog storage system.

Our solution is nontrivial because previous server-side deduplications with interactive key agreement require strong assumption that at least one end user, who has previously uploaded the same content, is supposed to be always online. Server-side deduplication without key agreement generally cannot allow the semi-honest fog storage to perform deduplication because it is difficult to identify duplication without knowledge of the encryption key.

The rest of this paper is organized as follows. In Section 2, previous secure deduplication studies are briefly reviewed. In Section 3, our goal for fog storage systems is defined. In Section 4, the proposed deduplication protocol is presented. In Sections 5 and 6, our scheme is compared and analyzed with previous deduplication approaches. In Section 7, the paper concludes with a summary of the proposed scheme.

II. Related work

Extensive researches on secure deduplication have been conducted recently under the consideration of cloud storage environments not specific to fog storage environments. Notice that there have been numerous secure deduplication protocols including what allows end users to outsource plaintext [11], [12], [13], [14] under the assumption that the remote storage service provider is fully trusted. Though, we assume that all of the remote storage cannot be fully trusted for the rest of this paper [8]. Thus, we focus on previous secure deduplication schemes which exploit cryptographic primitives at the end user side as regards data privacy. In this section, we briefly summarize some representative works and point out their limitations and difficulties in direct application to the upcoming fog storage architecture.

A. Client-side secure deduplication

Client-side deduplication occurs from the side of data-uploading entity (*i.e.*, end users rendered by IoT devices). Specifically, a client who attempts to upload data computes and sends a duplicate identification term (*e.g.* hash value) of the data to remote storage before actual outsourcing. When duplicate copy is discovered in remote storage, the upload requestor proves his ownership to the storage with modest amount of communications instead of uploading the entire content. By allowing only unique content to be outsourced to the remote storage, bandwidth consumption might diminish significantly and the effects of storage savings appear immediately.

Douceur *et al.* [10] introduced a cryptographic primitive, namely CE, by attempting to link data confidentiality via encryption with data deduplication. In this study, an encryption key is derived from the outsourced data in a deterministic way (*i.e.*, hash value) so that ciphertext generated under this key becomes the same. As a result, clients having the same content produce the same ciphertext without prior key agreement and the remote storage is allowed to identify duplicates via simple equality test without knowledge of encryption keys. Following this concept of deterministic key derivation from the data itself, Bellare *et al.* [15] presented a generalized framework, called message-locked encryption (MLE). MLE is categorized into four particular constructions and assessed rigorously according to levels of integrity and security guarantees.

However, some vulnerabilities arise in client-side deduplication approaches such as confirmation-of-file (CoF). CoF is indicated in [16], [17] as side information such that the knowledge of duplicates in the remote storage is revealed to adversaries. This might cause serious impact on data privacy when the message space is restricted to predictable space. Unfortunately, this side channel is inevitable because the identification of deduplicate content in the remote storage is a requisite component in client-side deduplication.

B. Server-side secure deduplication

As opposed to client-side deduplication, server-side deduplication occurs from the side of data-storing entity (*i.e.*, cloud and fog storage). The storage server receives all of the uploaded ciphertexts from clients, and then performs deduplication off-line or in the background during service provisioning. In a sense that the uploading client cannot be aware of duplicate copies in remote storage, this approach can be seen more secure than the client-side one.

Following the security vulnerabilities via side channels identified by Harnik *et al.* [11], Bellare *et al.* presented a server-aided deduplication, DupLESS [18]. This protocol adopts oblivious transfer protocol so that data owners of the same content can agree on pseudorandom encryption key. With a support of additional independent key server, it prevents adversaries from guessing plain content by exploiting the common hash value as side information. In addition, online brute-force attack can be effectively mitigated by rendering the key server as a rate-limiting authority.

Recently, Liu *et al.* proposed a server-side deduplication without additional independent servers [19] by exploiting password authenticated key exchange (PAKE) protocol. In this scheme, data-uploading user engages in key agreement protocol with online users who previously uploaded with the same short hash permitting collisions intentionally. During the process of PAKE protocol, communications pass through the cloud while secret values risking the data privacy are not transmitted. Once an agreed key is produced through PAKE protocol, the data-uploading user retrieves the encryption key and produces the same ciphertext stored in the cloud. This protocol is desirable in that neither side information is leaked to illegitimate users nor additional overheads are required in order to maintain additional servers. But network traffic is concentrated on the cloud, which is likely to cause a single point of failure. In addition, the assumption, that there should be at least one online user who has previously outsourced the same content, is too strong in pragmatic cloud storage environments.

Although the server-side deduplication achieves higher level of security than the client-side deduplication in terms of blockading leakage of side information, it requires more communications overheads than the one of client-side counterpart. This can be seen as a trade-off between data privacy and bandwidth efficiency, but efficient utilization of the network bandwidth becomes a crucial goal as the volume of outsourced data increases in the era of fog computing.

Besides, Stanek *et al.* [20] proposed an interesting idea considering both of client- and server-side deduplications based on the popularity of outsourced content. When it comes to popular outsourced data, existence of outsourced data can be leaked to adversaries, which brings about the same side channel vulnerabilities because then the encryption of popular content converges to the convergent encryption. The existence of trusted third party incurs additional maintenance overhead as well in this scheme.

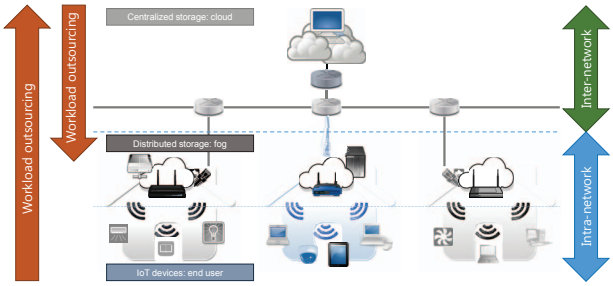


Figure 1: Architecture of a fog storage system

III. Fog storage description and our goals

In this section, we describe the architecture of fog storage systems and then define our goal of this paper.

A. Fog storage architecture

We introduce the three system entities in a typical fog storage system: the cloud, fog, and end user (Fig. 1).

1) The **cloud** is a centralized service provider which provides long-term data storage and retrieval services. The cloud maintains deduplicated data and metadata in the form of (data owner's ID, physical link to the data). In order to handle enormous volume of data and to avoid network congestion caused by its finite storage and network resources, its data storage workload is outsourced to widely distributed fog devices.
2) The **fog** is a distributed entity which is located at the edge of the network and provides data storage services in place of the cloud with its own limited resources. Fog devices are connected to the central cloud and also possibly with each other (via inter-network), while it is connected to end-user devices in a spatially restricted domain (via intra-network). It is responsible for temporal storage services and relays services from the cloud when temporal/spatial workloads exceed its capability.
3) The **end user** is the data outsourcing/retrieving entity (*e.g.* IoT devices). The principal goal of end users is to receive data storage/retrieval services with elasticity and scalability from the cloud storage through the fog storage. They have relatively very limited amount of storage space compared to the one of cloud and fog storage. Thus, we assume that they remove the data content from their local storage after outsourcing it.

B. Efficiency requirements

Taking into account the fog storage architecture explained above, bandwidth consumption needs to be minimized in inter-network communication. In other words, heavy network traffic is likely to be concentrated to the central cloud

because it should receive uploaded data from several fog storage and transmit stored data to them in order to maintain the overall storage services. Therefore, to control network bandwidth effectively, it is desirable to perform client-side deduplication between the cloud storage and the fog storage. In practice, the shift of heavy network load to distributed small area domains is natural because network condition between the fog storage and end users is expected to be better than the former because of its physical proximity. As long as the impact of server-side deduplication is negligible, it is admissible to perform server-side deduplication between the fog storage and end users.

C. Adversarial model

In a fog storage system, we assume that the cloud and fog devices are honest-but-curious entities. To clarify an adversary's location and capabilities, it can be broadly categorized into one of the two groups: inside and outside.

- The **inside adversary** participates in the data outsourcing services and follows the prescribed protocols. At the same time, it attempts to learn information about the underlying plain content of the outsourced data by exploiting transcripts allowably given to it. The cloud and fog storage belong to this kind of adversary and end users who have not outsourced data to them can be considered potential inside adversaries. Therefore, it can be regarded as passive attacker.
- The **outside adversary** attempts to acquire the underlying plain data without obedience of the given protocol. It can exploit side channels such as eavesdropping, access and possibly modify physical storage in an inappropriate manner. Therefore, it can be considered active attacker.

Without loss of generality, we assume that the cloud and fog do not collude with other adversaries so that they cannot learn about existence of duplicates in remote storage.

D. Security requirements

Taking the aforementioned adversaries into account, the following properties should be satisfied in the proposed scheme:

1) Confidentiality: Unauthorized access to plain data should be prevented to deter from unintended exposure of outsourced data.
2) Integrity: End users should be able to verify the data integrity after retrieval of their outsourced content.
3) Leakage resilience: Information leakage should be minimized during data outsourcing process. This differs from confidentiality in that no side information should be given to even valid end users who have outsourced the data content to the remote storage.

IV. Proposed hybrid secure deduplication

In this section, the proposed hybrid deduplication scheme in fog storage architecture is described.

A. Preliminaries

Prior to a detailed description of the proposed scheme, the cryptographic primitives on which it is based will be briefly summarized.

1) Bilinear maps: Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups with a large prime order p where g is a generator of $\mathbb{G}$. Then, there is an efficiently computable bilinear map e such that $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ holds for any $u, v \in \mathbb{G}$ and any $a, b \in \mathbb{Z}_p$.
- Non-degeneracy: $e(g, g) \neq 1$.

2) Bilinear Diffie-Hellman assumption: The bilinear Diffie-Hellman (BDH) problem is to compute $e(g, g)^{abc} \in \mathbb{G}_T$ given $\langle g, g^a, g^b, g^c \rangle$ for randomly chosen values $a, b, c \in_R \mathbb{Z}_p$. Let $Adv_{\mathcal{A}}^{BDH}$ be the advantage that any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ solves the BDH problem as follows

$$Adv_{\mathcal{A}}^{BDH} = Pr[\mathcal{A}(e, g, g^a, g^b, g^c) = e(g, g)^{abc}].$$

If $Adv_{\mathcal{A}}^{BDH}$ is a negligible function in the security parameter λ, then we can say that BDH assumption holds and denote by $\langle p, \mathbb{G}, \mathbb{G}_T, g, e \rangle$ the BDH group.

3) Identity-based encryption (IBE): Waters introduced an efficient IBE [21] without random oracles. The Waters' IBE is comprised of the following three probabilistic algorithms and one deterministic algorithm $(Setup, KeyGen, Enc, Dec)$.

- $(\mu, mk) \xleftarrow{\$} Setup(1^\lambda)$: Given the security parameter λ, choose a random BDH group $\langle p, \mathbb{G}, \mathbb{G}_T, g, e \rangle$ as defined above. In addition, choose random values $\alpha \in_R \mathbb{Z}_p$, $g_2 \in_R \mathbb{G}$, and $u', u_1, \ldots, u_N \in_R \mathbb{G}$ for some integer $N \in \mathbb{N}$. Define a pseudorandom function $F(v) = u' \prod_{i \in \mathcal{V}} u_i$ where v is N-bit identity string and $\mathcal{V}$ is a set of indices on which the value is 1 in v. The public parameter then becomes $\mu = (g, g_1 = g^\alpha, g_2, F(\cdot))$ and the master secret key becomes $mk = g_2^\alpha$.
- $d_v \xleftarrow{\$} KeyGen(\mu, mk, v)$: Given N-bit identity, choose a random value $r \in_R \mathbb{Z}_p$. The decryption key for identity v then becomes $d_v = (mk \cdot F(v)^r, g^r) = (g_2^\alpha \cdot F(v)^r, g^r)$.
- $C \xleftarrow{\$} Enc(\mu, v, M)$: For a message M and identity bitstring v, choose a random value $t \in_R \mathbb{Z}_p$. The ciphertext then becomes $C = (M \cdot e(g_1, g_2)^t, g^t, F(v)^t)$.
- $M \leftarrow Dec(\mu, d_v, C)$: For decryption key $d_v = (d_1, d_2)$ and ciphertext $C = (c_1, c_2, c_3)$, the plaintext becomes $M = c_1 \cdot e(d_2, c_3)/e(d_1, c_2)$.

Its security is based on decisional BDH assumption and the detailed proof can be found in [21]. In the proposed scheme, slightly modified version of Waters' IBE is used where the identity is represented as pseudorandom value

generated from the message to be outsourced.[1]

B. Overview

Once the system public parameters are established, end users are allowed to transfer their content to a fog storage in charge and retrieve the content from it on demand.

Each end user outsources a randomized encryption of his data content by choosing random exponent and deletes both plaintext and ciphertext from his local storage. The fog storage, located on user side, then stores the received ciphertext for a certain period of time. During this period of time, the fog storage performs server-side deduplication locally when an encryption of the same content is uploaded by end users. Upon times excess or storage overload, the fog storage and the central cloud storage engage in client-side deduplication. If encryptions of the same content are stored in distributed fog storage, the central cloud receives just (random) one of them and all of the fog storage remove the encryption from its temporal storage.

To access the outsourced data, the end user receives possibly deduplicated ciphertext from the fog device in charge. At this time, the fog device relays the ciphertext belonging to the end user from the central cloud to the end user if it is not stored in its local storage. Otherwise, the fog directly transfers the corresponding ciphertext to the end user.

C. Main construction

From now on, we describe the entire process of our deduplication protocol. This procedure is composed of four phases ($System\ setup$, $Data\ outsourcing$, $Retrieval$, $Deduplication$). It is based on Waters' IBE scheme, but is modified version in order to enable secure deduplication. The specific algorithms are described in Algorithm 1.

1) System setup: The trust initializer runs the $Setup$ algorithm to obtain system public parameters spp. spp is disclosed to the public so as to allow any entity who engage in the data outsourcing to perform the prescribed protocol.

2) Data outsourcing: When an end user tries to upload content to the fog storage, he first generates two pseudo-random values derived from the content, which are used in encryption (as an identity of IBE scheme). By using these values, he generates a random encryption and decryption key in the $Encrypt$ algorithm. He then transmits the ciphertext to the fog storage, which locates at the same or near network of the end user. The decryption key dk and hash digest α are kept secret by the end user and used to access the outsourced content and integrity verification, respectively. Without loss of generality, he removes all of the parameters except the decryption key for saving storage space. It is notable that the end user who outsources his encrypted content cannot learn any information including existence of the same content in the fog storage.

3) Retrieval: When an end user wants to access his previously outsourced content from the fog storage, he sends a retrieval request to it. If the fog storage still stores an encryption of the user's content, it just sends the ciphertext to the end user. Otherwise, the fog storage forwards retrieval request to the central cloud storage and conveys the received ciphertext to the end user. The central cloud storage always stores only single copy (as an encryption) of the data content outsourced by end users, so it is able to deliver the requested encryption to the fog storage. The end user then runs the $Decrypt$ algorithm to obtain the plain content.

4) Deduplication: Once a fog storage receives ciphertext C from an end user, it checks duplicates by running $Dedup$ algorithm with another ciphertext C' stored in its local storage. If duplicate copy is found, the fog storage replaces previously stored ciphertext with the new ciphertext received just before for the same content and registers the end user as valid data owner (*i.e.*, server-side deduplication).[2] Otherwise, the fog storage requests (periodically or on demand) duplicate check to the central cloud storage by sending just last two parameters in the received ciphertext. The central cloud runs the $Dedup$ algorithm as the fog did, but it returns the result of the algorithm to the fog. If the result is $False$ (in case of unique data not stored in the cloud), the fog forwards the first component c_1 and the cloud stores it in the form $C = (c_1, c_2, c_3)$ (*i.e.*, client-side deduplication). Regardless of the result, the fog removes the ciphertext from its local storage at the end of $Dedup$ algorithm.

V. Performance analysis

In this section, the proposed scheme is analyzed and compared with previous secure deduplication approaches in terms of efficiency.

A. Implementation

In order to evaluate the efficiency of the proposed scheme, prototypes of ours, Bellare *et al.*'s scheme [18], and Liu *et al.*'s scheme [19] are implemented in Python (version 3.4.3) upon Charm framework [23]. Then, we evaluate the performance with average values over 100 times repetitions on a desktop PC running Ubuntu 12.04 LTS with 3.4GHz CPU with 16GB RAM. In order to follow NIST recommendation with 256-bit security [24], secure symmetric encryption/decryption is rendered by AES256, universal one-way hash functions are implemented by exploiting SHA512 and SHA256, and supersingular elliptic curve with 512-bit base field is used in the prototypes.

[1]Specifically, the master secret $mk = g_2^{\alpha}$ and g_1 in public parameter spp are removed from the proposed protocol. This removes the necessity of trusted key generation center (note that the cloud and fog cannot be fully trusted). Fortunately, the secret key can be generated by valid data owners from the message, which is similar to the Boneh-Boyen IBE protocol [22].

[2]This duplicate check can be done by fog storage in the background at any time. The selection criterion which copy should be kept is dependent on a policy of the fog storage, but it has no effect on the security of the scheme.

Algorithm 1 Algorithms in the proposed scheme

procedure $Setup(1^\lambda)$ ▷ System public parameter selection
Select a random BDH group $\langle p, \mathbb{G}, \mathbb{G}_T, g, e\rangle$
Select uniform random values $g_2, u', u_1, u_2, \ldots, u_N$ from $\mathbb{G}$ for some $N \in \mathbb{N}$
Define a functiton $F(s) = u' \prod_{i \in V} u_i$ given N-bit string s with a bitmap V for which i-th bit of s is 1
Select semantically secure symmetric encryption/decryption with m-bit key ek for some $m \in \mathbb{N}$ such that
$C \leftarrow SE(ek, M)$ and $M \leftarrow SD(ek, C)$
Select universal one-way hash functions $\mathcal{H}_1, \mathcal{H}_2$, and $\mathcal{H}_3$ such that
$\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$, $\mathcal{H}_2 : \{0,1\}^* \rightarrow \{0,1\}^N$, $\mathcal{H}_3 : \mathbb{G}_T \rightarrow \{0,1\}^m$
return $spp = \langle p, \mathbb{G}, \mathbb{G}_T, g, g_2, e, F(\cdot), \mathcal{H}_1(\cdot), \mathcal{H}_2(\cdot), \mathcal{H}_3\rangle$
end procedure

procedure $SigGen(spp, M)$ ▷ Pseudorandom value generation
$\alpha \leftarrow \mathcal{H}_1(M)$
$\sigma_{M,1} \leftarrow g_2^\alpha$
$\sigma_{M,2} \leftarrow F(\mathcal{H}_2(M))$
return $\sigma = \langle \alpha, \sigma_{M,1}, \sigma_{M,2}\rangle$
end procedure

procedure $Dedup(spp, \tilde{C} = (\cdot, c_2, c_3), \tilde{C}' = (\cdot, c'_2, c'_3))$ ▷ (Client-side) Duplicate identification
if $e(c_2, c'_3) = e(c'_2, c_3)$ **then**
return $True$
end if
return $False$
end procedure

procedure $Encrypt(spp, \sigma, M)$ ▷ Probabilistic encryption
Select uniform random values $r, t \in_R \mathbb{Z}_p$
$dk \leftarrow (g_2^\alpha \cdot \sigma_{M,2}^r, g^r)$
$ek \leftarrow \mathcal{H}_3(e(\sigma_{M,1}, g_2)^t)$
$C \leftarrow (SE(ek, M), g^t, \sigma_{M,2}^t)$
return $\langle dk, C\rangle$
end procedure

procedure $Decrypt(spp, dk, C)$ ▷ Deterministic decryption
$(dk_1, dk_2) \leftarrow dk$
$(c_1, c_2, c_3) \leftarrow C$
$ek \leftarrow \mathcal{H}_3(e(dk_1, c_2)/e(dk_2, c_3))$
$M \leftarrow SD(ek, c_1)$
return M
end procedure

B. Computation overheads

Fig. 2 demonstrates computation time (ms) on client side. Per algorithm time consumptions are depicted in bar graph and total time for single data outsourcing is drawn with a solid line. In all evaluations, computation time for file I/O, encryption, and decryption are almost the same with standard deviation less than 0.001 due to the same cryptographic encryption upon the same machine. In addition, all schemes have similar computation time dominated by the encryption.

Oblivious pseudorandom key generation (denoted by OPRF) in Bellare *et al.*'s scheme is performed on the hash value of outsourcing content and requires 1 exponentiation and 2 multiplications, while $SigGen$ algorithm in the proposed scheme uses two universal hash functions and constant number of multiplications (*i.e.*, 257 multiplications in $F(\cdot)$ evaluation of prototype) and 1 exponentiation.[3]

On the contrary, Liu *et al.*'s scheme requires constant number of hash (for both of full-length and short) and key derivation, but repetitive computation of PAKE in proportion to the number of online checkers who have the same short hash value. Thus, computation time increases as the number of checkers increases.

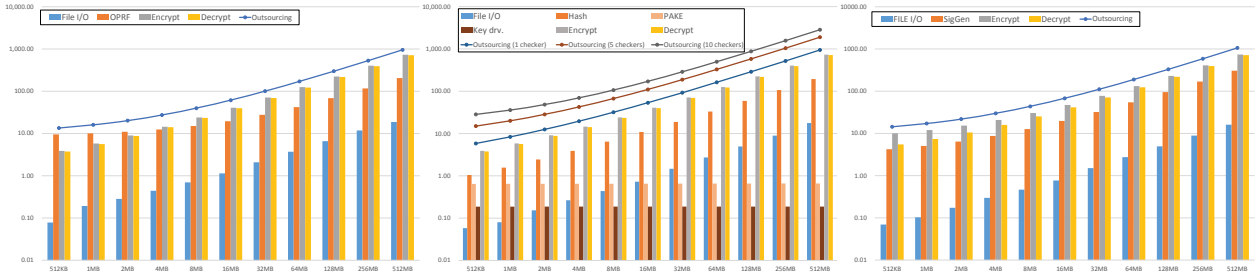
The other constant times are summarized in TABLE I. Although the proposed scheme requires almost 3 times more computation time for system setup, this can be done only once before service provisioning. Liu *et al.*'s setup time is minimal because the system does not require initial setup of parameters for neither elliptic curve nor RSA.

Unlike the proposed scheme, Bellare *et al.*'s scheme and Liu *et al.*'s scheme require computation on server side during data outsourcing. In Bellare *et al.*'s DupLESS, the key server needs to engage in oblivious encryption key generation. In Liu *et al.*'s scheme, the central server is supposed to send data outsourcing entity homomorphically encrypted masked encryption key. When there is no duplicate encryption in the server, the server randomly chooses the key without homomorphic addition, which causes reduction of 167μs in key generation. Additionally, Liu *et al.*'s scheme requires multiple online checkers to participate in PAKE protocol for the same short hash value.

C. Storage overheads

After data outsourcing, data owners need to keep the decryption (*i.e.*, encryption) key secret. Both of Bellare *et*

[3]This makes difference of ours from Bellare *et al.*'s scheme in that ours is 2.25 times faster for 2MB data but 0.67 times for 512MB (converging to 35% slower than Bellare *et al.*'s scheme for data larger than 512MB). But it is less than half of encryption time making the total computation time in outsourcing negligible.



(a) Bellare et al. (b) Liu et al. (c) Proposed scheme

Figure 2: Computation time (ms) on client side with varying data size

Table I: Constant computation time in data outsourcing

Entity	Phase (algorithm)	Scheme	Time (ms)
Trusted initializer	Setup	Bellare *et al.*	119.2147
		Liu *et al.*	2.5296
		Proposed	370.5741
(Key) Server	OPRF	Bellare *et al.*	4.1141
	Key combine (Dedup.)	Liu *et al.*	0.2852
	Key gen. (Non-dedup.)	Liu *et al.*	0.2685
Online checker	PAKE	Liu *et al.*	0.3759

al.'s scheme and Liu *et al.*'s scheme require the owners to store (256-bit) AES encryption key. Instead of directly storing the key, the proposed scheme allows data owners to keep the key derivation key. Although this mechanism incurs additional overhead by introducing two elliptic curve points (in dk), it enables independent data owner to choose his random encryption key without additional interactive communications for key agreement or derivation.

On the server side, outsourced encrypted contents along with metadata related to data owners need to be maintained. Focusing on the ciphertext, the proposed scheme includes two additional elliptic curve points (*i.e.*, c_2 and c_3) which might be negligible as the size of outsourced content becomes larger.[4]

D. Communication overheads

In Liu *et al.*'s protocol [19], every communication passes through the central server (*i.e.*, cloud). On the other hand, end users interact with two independent servers in Bellare *et al.*'s protocol [18]: one with the key server for oblivious key derivation and the other with cloud storage for data outsourcing. In any case, all the communications of above studies are transmitted at inter-network level.

On the contrary, the proposed scheme considers both inter-network and intra-network communications in a single secure deduplication protocol. Therefore, heavy traffic conveying the ciphertext mostly occurs in a restricted domain via intra-network communication.

In order to evaluate the bandwidth efficiency, we ran micro benchmarks with varying intra-network conditions.[5] Fig. 3 depicts transmission time for outsourcing a single 512MB content with different intra-network speed. Without loss of generality, as the transmission time is bounded on slower network, the maximum transmission speed follows that of inter-network in the previous approaches. This implies that network traffic can be concentrated to the network core even when most of the traffic can be handled in the network edge such that communication condition between the end users and fog is superior than that between the fog and cloud. Notably, the proposed scheme can transmit much faster than others as long as the intra-network speed is faster than the inter-network.

As the size of outsourced data becomes larger, differences of communication costs among previous approaches seem negligible. As it becomes smaller, however, differences of them would be larger as depicted in Fig. 4. As long as the condition of inter-network is better, all the scheme shows similar pattern. In Liu *et al.*'s approach, communication overhead becomes larger as the number of online checkers increases (owing to high probability of short hash value). Under any circumstances, the proposed scheme seems to most effectively utilize the network bandwidth.

VI. Security Analysis

A. Confidentiality

When adversaries obtain outsourced ciphertext in an illicit way, they cannot learn any information about plain content from the ciphertext. The main difference of our protocol from Waters' IBE is that the master secret is generated by valid data owners rather than by the trusted key generation center. Nevertheless, we give a brief sketch of security proof of the proposed protocol that is bounded on the Waters' IBE.

[4]This additional overhead occupies less than 0.035% even for 512KB data in the storage server. In Charm framework, an element in the pairing module with 512-bit base field is serialized to a 90-byte stream.

[5]In this simulation, we consider the network condition as goodput. According to average Internet speed [25], we set the goodput of inter-network 20Mbps.

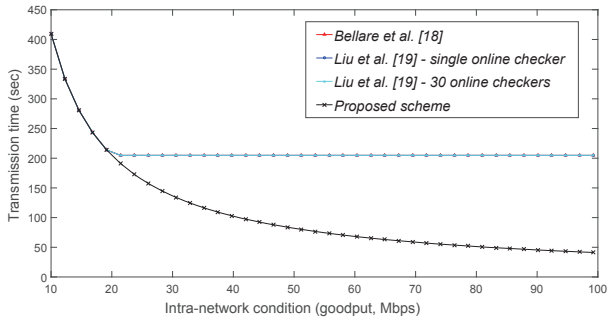


Figure 3: 512MB data transmission in 20Mbps inter-network

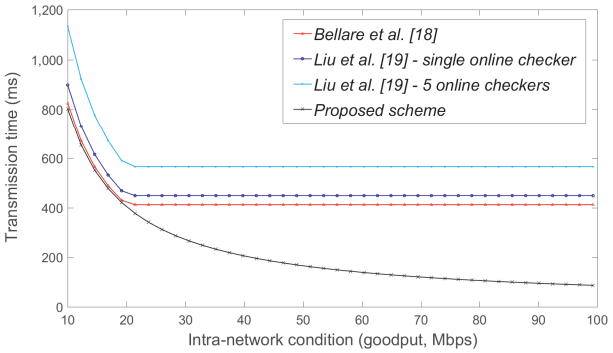


Figure 4: 10KB data transmission in 20Mbps inter-network

Proposition 1. *As long as the symmetric encryption is semantically secure and universal hash functions satisfy one-wayness, our scheme is as secure as Waters' IBE [21].*

Proof: By means of eavesdropping, any (insider or outside) adversary without the entire data can obtain $\{g, g_2\} \subset spp$ and $\{g^t, \sigma_{M,2}^t\} \subset C$. Without loss of generality, we can set $\alpha = \mathcal{H}_1(M), g_2 = g^\beta, t = \gamma, \sigma_{M,2} = g^\delta$. Actually, α and δ are dependent on message, but according to hardness assumptions of discrete logarithm (resp. $\sigma_{M,1} = g^{\mathcal{H}_1(M)}$) and subset product (resp. $\sigma_{M,2} = F(\mathcal{H}_2(M))$), α and δ look independent and random from the perspective of adversaries without knowledge of the entire data M. Therefore, the adversaries regard $\langle \alpha, \beta, \gamma, \delta \rangle$ as independent and random values.

Now, suppose that the PPT adversary $\mathcal{B}$ can break the proposed scheme with non-negligible probability. We can build another PPT adversary $\mathcal{A}'$ that can break the Waters' IBE by exploiting $\mathcal{B}$. Given $\langle g, g^\alpha, g^\beta, g^\delta, g^{\gamma\delta} \rangle$, $\mathcal{B}$ can compute $e(g,g)^{\alpha\beta\gamma}$ by the assumption. Accordingly, the adversary $\mathcal{A}'$ taking the same challenge (*i.e.*, $\langle g, g^\alpha, g^\beta, g^\delta, g^{\gamma\delta} \rangle$) as input invokes the adversary $\mathcal{B}$ with the challenge, obtains $e(g,g)^{\alpha\beta\gamma}$ as a result of $\mathcal{B}$, and learns the plain data M by removing masking factor $e(g,g)^{\alpha\beta\gamma}$ from the ciphertext $M \cdot e(g,g)^{\alpha\beta\gamma}$ through simple division. It contradicts the security assumption of Waters' IBE. Thus, the proposed scheme is as secure as Waters' IBE under bilinear Diffie-Hellman assumption. ■

B. Integrity

When valid data owner obtains outsourced plain data M via $Retrieval$ phase, he can verify integrity of his restored data through equality test of $\mathcal{H}_1(M)$ with the stored value α.

C. Leakage resilience

In the proposed scheme, end users trying to outsource their contents cannot learn existence of encryptions for the same content. This deters adversaries from launching confirmation-of-file attack, which is devised to reveal existence of upload-requesting content in the remote storage by analyzing the response from the remote storage in client-side deduplication. In addition, duplicate identification is not conducted by simple comparison of relatively small values (*i.e.*, hash values) in the proposed scheme. By allowing each end user to select a uniform random value r in $Encrypt$ algorithm, adversaries without valid decryption key cannot learn any information during $Data\ outsourcing$ phase.

VII. Conclusion

Ever-increasing volume of data induced data outsourcing to become popular. With privacy concerns in conjunction with efficient resource management, remote storage service providers are supposed to provide a secure and efficient way to manage outsourced contents. In this paper, we presented a simple but secure data deduplication protocol with best-effort bandwidth efficiency. Adaptive combination of client- and server-side deduplication according to network condition makes deduplication more eligible for environments with tremendous amount of outsourced data. Moreover, this hybrid approach effectively eliminates leakage of side information enabling more reliable data outsourcing. Efficiency and security analyses demonstrate that the proposed approach is well suited to the upcoming fog computing era with data explosion.

We plan to extend our research through evaluation under real environments by deploying the prototypes in Android mobile devices (which is supported by Charm framework) and by measuring real time consumption upon mobile devices and transmission time. We also take into account an optimization of the proposed scheme including efficiency-enhancement of duplicate identification as well as security-enhancement considering offline brute-force attack on remote storage side.

Acknowledgment

This work was supported by a Korea University Grant. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No.2016R1A2A2A05005402). This work was also

supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.B0717-16-0128, Research on security model for fog computing platform).

REFERENCES

[1] R. van der Meulen, "Gartner says 6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015," *http://www.gartner.com/newsroom/id/3165317* (11/10/2015).

[2] IDC Market in a Minute: Internet of Things, *http://www.idc.com/downloads/idc_market_in_a_minute_iot_infographic.pdf*.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *ACM Workshop on Mobile Big Data*, 2015, pp. 37–42.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC Workshop on Mobile Cloud Computing*, ACM, 2012, pp. 13–16.

[5] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, Nov 2014, pp. 117–122.

[6] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer International Publishing, 2014, pp. 169–186.

[7] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2014, pp. 325–329.

[8] "Datalossdb," *http://datalossdb.org*, Open Security Foundation.

[9] Verizon, "2015 data breach investigation report," *http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigation-report-2015_en_xg.pdf*.

[10] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," *International Conference on Distributed Computing Systems (ICDCS)*, 2002, pp. 617–624.

[11] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 40– 47, Nov 2010.

[12] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," *ACM Conference on Computer and Communications Security*. 2011, pp. 491–500.

[13] J. Barreto, L. Veiga, and P. Ferreira, "Hash challenges: Stretching the limits of compare-by- hash in distributed data deduplication," *Information Processing Letters*, vol. 112, no. 10, pp. 380 – 385, 2012.

[14] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication,"
emphACM Conference on Data and Application Security and Privacy. 2012, pp. 1–12.

[15] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-Locked Encryption and Secure Deduplication," *Advances in Cryptology – EUROCRYPT*, Springer Berlin Heidelberg, Athens, Greece, May 26-30, 2013, pp. 296–312.

[16] P. Puzio, R. Molva, M. Onen, and S. Loureiro, "Cloudedup: Secure deduplication with encrypted data for cloud storage," *International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2013, pp. 363– 370.

[17] M. Wen, S. Yu, J. Li, H. Li, and K. Lu, "Big Data Storage Security," *Big Data Concepts, Theories, and Applications*. Springer International Publishing, 2016, pp. 237–255.

[18] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," *USENIX Security Symposium (USENIX Security)*. 2013, pp. 179–194.

[19] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," *ACM Conference on Computer and Communications Security*. 2015, pp. 874–885.

[20] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A Secure Data Deduplication Scheme for Cloud Storage," *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, March 2014, pp. 99–118.

[21] B. Waters, "Efficient Identity-Based Encryption Without Random Oracles," *Advances in Cryptology – EUROCRYPT*, Springer Berlin Heidelberg, May 2005, pp. 114–127.

[22] D. Boneh and X. Boyen, "Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles," *Advances in Cryptology – EUROCRYPT*, Springer Berlin Heidelberg, 2004, pp. 223–238.

[23] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[24] "Cryptographic key length recommendation," *https://www.keylength.com/en/4/*.

[25] "List of countries by internet connection speeds," *https://en.wikipedia.org/wiki/List_of_countries_by_Internet_connection_speeds*.