International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 07 | July 2020 www.irjet.net p-ISSN: 2395-0072

Secure Deduplication Scheme on Big Data Storage of Cloud

Pruthveesh C [1], Renuka prasad R[2], Divakar K M [3]

[1]*Student, Dept. of CSE, S J C Institute of Technology, Chickballapur, Karnataka, India*
[2]*Student, Dept. of CSE, S J C Institute of Technology, Chickballapur, Karnataka, India*
[3]*Assistant Professor, Dept. of CSE, S J C Institute of Technology, Chickballapur, Karnataka, India*

Abstract - *Secure data deduplication as it can eliminate redundancies over encrypted data, has been widely developed in cloud storage to reduce storage space and communication overheads. Among them, the convergent encryption has been extensively adopted. However, it is vulnerable to brute-force attacks that can determine which plaintext in a message space corresponds to a given ciphertext. Many existing schemes have to sacrifice efficiency to resist brute-force attacks, especially for cross-domain deduplication, which is inevitably contrary to practical applications. Moreover, few existing schemes consider protecting the message equality information .we propose an efficient and privacy-preserving big data deduplication scheme for a two-level multi-domain architecture. Specifically, by generating a random tag and a constant number of random ciphertexts for each data.*

Key Words: Secure data deduplication, cross-domain deduplication, brute-force attacks, message equality information.

1. INTRODUCTION

UNDER big data-driven society, data deduplication technique has been widely developed in cloud storage because it can significantly reduce storage costs by storing only a single copy of redundant data. Indeed, data deduplication can reduce storage costs by more than 50% in standard file systems and by more than 90% for backup applications, and these savings translate into substantial financial savings to cloud service providers and users [2]. However, considering security and privacy concerns of outsourced data, users are likely to encrypt data with their own keys before outsourcing. This impedes the cross-user deduplication since an identical data will be encrypted into different ciphertexts by different users' keys, i.e., it is challenging to identify duplicates over different ciphertexts. Thus, how to efficiently conduct data deduplication over encrypted data becomes a pressing issue.

Convergent encryption [3] provides the first viable solution to allow deduplication on encrypted data, which is formalized as the message-locked encryption later in [4]. It encrypts the data with the convergent key derived by computing the cryptographic hash value of the data itself. Since the encryption is a deterministic symmetric encryption algorithm, identical data will generate the same convergent key and ciphertext, which achieves deduplication on encrypted data. However, the convergent encryption is vulnerable to brute-force attacks due to its deterministic property. To resist such attacks, some server-aided encryption schemes [5], [6], [7] have been presented. In these schemes, a domain or tenant (e.g., a company or university) deploys a dedicated key server to help affiliated users to generate the convergent key and the corresponding tag through an interactive protocol. Unfortunately, since key servers in different domains randomly select different secret keys, cross-domain deduplication is infeasible, which dramatically reduces the effectiveness of data deduplication.

2. RELATED WORKS

2.1 support cross-domain deduplication, an inter-deduplication algorithm:

Since only the encrypted data uploaded by the first user will be stored in the cloud, to ensure that the encrypted data can be correctly decrypted by users with ownership, the number of generated ciphertexts is linear with the number of domains. Thus, the scalability issue will inevitably arise when the number of domains explodes, which will lead to massive computational and communication overheads for users and the cloud service provider. Moreover, the message equality information of outsourced data (i.e., the information about whether two different ciphertexts correspond to an identical plaintext) may also leak the content of encrypted data to some extent [2]. Although this information disclosure is inevitable in data deduplication, we hope to minimize such information leakage. Recently, two solutions [9], [10] have been drawn to achieve this goal. The crux of both schemes is to permit only the deduplication operator to know such information.

2.2 deduplication operator

The first scheme [9] employs an additional trusted server to complete the duplicate verification. However, the assumption that the trusted server is always online and will not be compromised is too strong to be accepted in practice [11]. The second scheme [10] seems a good candidate because it protects the message equality information by introducing a technical method rather than an additional trusted server. Besides, this scheme efficiently achieves cross-domain deduplication and resists brute-force attacks

at the same time. Unfortunately, this scheme can only support the crossdomain deduplication for two different domains. secure cross-domain (or cross-user) deduplication schemes are evaluated in terms of two factors: security properties that each scheme provides and system overheads incurred on the cloud service provider and users.existing related schemes cannot efficiently achieve the strong confidentiality of outsourced data while resisting bruteforce attacks.

2.3 we propose an efficient and privacy-preserving multi-domain big data deduplication scheme in cloud storage:

There are three contributions that this paper gives:

- First, we propose a secure multi-domain deduplication scheme that can support data deduplication not only in the same domain (called intra-deduplication) but also across multiple different domains (called inter-deduplication). Specifically, the proposed scheme generates the random convergent key and the random tag based on the bilinear pairing technique [13] to ensure data confidentiality. The proposed scheme only needs to produce a constant number of random ciphertexts to ensure that the outsourced encrypted data can be correctly decrypted by users with ownership. In the proposed scheme the cloud service provoider performs interdeduplication using Boneh-Goh-Nissim cryptosystem [14].
- Second, to improve the time complexity of duplicate search, we construct a deduplication decision tree based on the B+ tree [15], which works well for the big data storage system. In particular, the length of plaintext can be used to represent the keyword in the B+ tree.
- Third, we analyze the security of the proposed scheme and demonstrate that it can achieve strong data confidentiality and data integrity while resisting brute-force attacks. Besides, extensive performance evaluations justify the efficiency of the proposed scheme in terms of computational, communication and storage overheads.

2.4 System Model:

Two-level multi-domain deduplication model, which provides the intra-deduplication and inter-deduplication.
More specifically, the first level contains a number of domains D = {D1,D2,...,Dn}. Each domain Di corresponding to an organization

The second level includes a cloud service provider, which conducts the inter-deduplication

- Key distribution server (KDS): The KDS is responsible for generating different private keys for users from different domains and a secret for the cloud service provider to perform the inter-deduplication
- Cloud service provider (CSP): The CSP offers storage services for users. Although the CSP seems to have abundant storage space, the corresponding costs of the management and maintenance for big data are relatively expensive. Hence, the CSP prefers to conduct the inter-deduplication over outsourced data from all domains to save costs by storing only one copy
- Users: Users from the same domain receive the same private key from the KDS, on the contrary, users from different domains possess different private keys. Based on the received private key, users can generate a random convergent key used for encrypting the data and an intratag used for deduplicating.
- Agent (Ai): In order to improve the efficiency of duplication search and resist an online brute-force attack launched by malicious users, an agent Ai located between users and the CSP is hired by Di for performing intra-deduplication and transforming a random intra-tag into a random intertag.

2.5 A random intra-tag intradeduplication:

If a duplicate is found, Ai returns the corresponding feedback. Otherwise, Ai transforms the intra-tag into a random inter-tag and sends it to the CSP for inter-deduplication. Note that only when the CSP does not find a duplicate, U needs to encrypt m and upload the corresponding ciphertext.

If a duplicate is found, Ai returns the corresponding feedback. Otherwise, Ai transforms the intra-tag into a random inter-tag and sends it to the CSP for inter-deduplication. Note that only when the CSP does not find a duplicate, U needs to encrypt m and upload the corresponding ciphertext.

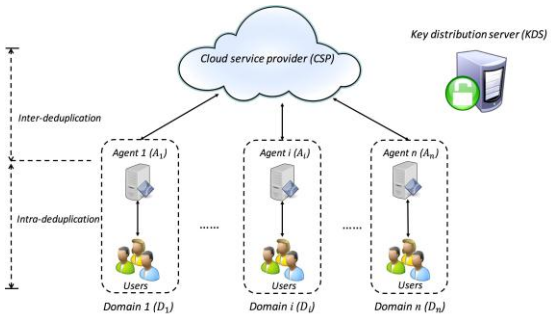


Fig. 1: System model under consideration

2.6 Threat Model:

In our system, the KDS is assumed to be completely trusted and would hardly be compromised by any adversary because it only participates in system setup. Similar to the existing literature dealing with the security in cloud storage [16], [17], the CSP and Ai, considered as honest-but-curious, will honestly follow the underlying scheme, but are curious about contents of outsourced encrypted data. Specifically, based on the background knowledge about the plaintext space, the CSP would collude with Ai to guess the content of a targeted ciphertext by launching an offline bruteforce attack. Concretely, they will not frame the CSP or corresponding Ai by uploading invalid data. However, they may be interested in obtaining the content of encrypted data without ownership. Besides, if a malicious user (corrupted by the adversary) knows the plaintext space, he/she would try to launch an online bruteforce attack by repeatedly executing the data upload procedure to guess the content of the targeted ciphertext. Note that users would not collude with the CSP or Ai due to their own privacy concerns and the reputation of the CSP or Ai

3. RELATED WORKS

Based on the threat model,

- Data confidentiality. There exist two kinds of data confidentiality in our model. One is related to the semantic security of encrypted data and tags, i.e., any adversary even corrupting the CSP and Ai or unauthorized users cannot feasibly extract any information about a plaintext from its ciphertext or tag.

- Data integrity. In the design of secure data deduplication, only one copy will be stored in the CSP. Since this unique copy may be altered due to some physical failures or monetary reasons, the proposed scheme should provide data integrity to ensure that users can verify whether the downloaded data are modified or not.

3.1 MODELS AND DESIGN GOALS:

We formalize the system model and threat model used in this paper, and identify our design goals.

- Brute-force attack resistance. With the background knowledge of the plaintext space, any adversary even corrupting the CSP or Ai cannot determine which plaintext corresponds to a specific tag and ciphertext through an offline brute-force attack. It is worth noting that the CSP or Ai can determine whether two ciphertexts correspond to the same plaintext by comparing received tags, but they cannot determine which plaintext in the plaintext space corresponds to these ciphertexts. Besides, the malicious user (corrupted by the adversary) tries to guess the content of the targeted ciphertext by launching online brute-force attacks. Thus, the proposed scheme should also try to resist such attacks.

- Efficiency. Apart from security requirements, efficiency is the most important metric for data deduplication [16]. Actually, applying secure deduplication would lead to some unavoidable costs for users and the CSP [12], e.g., tag generation, duplication search, and verification. Therefore, achieving the efficiency of computational, communication and storage overheads is also our design goal, especially in today's big data-driven society. Meanwhile, facing a vast volume of data, the time complexity of duplication search should be reduced as much as possible.

4. PRELIMINARIES

we outline the bilinear groups of composite order

4.1 Bilinear Groups of Composite Order

Given a security parameter κ, a composite bilinear parameter generator Gen(κ) outputs a tuple (p,q,G,GT,e), where p and q are two κ-bit primes, G and GT are two finite cyclic multiplicative groups of composite order N = pq, and e : G × G → GT is a bilinear map with the following properties:

- Bilinear: e(xa,yb) = e(x,y)ab for all x,y ∈ G, and a,b ∈ ZN.

- Non-degeneracy: If g is a generator of G, then e(g,g) is a generator of GT with the order N.
- Computability: For all x,y ∈ G, there exists an efficient algorithm to compute e(x,y) ∈ GT.

4.2 Intra-deduplication

"uploadk(Lm,τm)", Ai performs the intra-deduplication as follows.
1) Based on the private key ai, compute
(2)Then, compute the hash of .

4.2.1 Inter-deduplication

After receiving the message "uploadk(i,Lm,τˆm)" from Di, the CSP performs the inter-deduplication to further eliminate the redundancy of data. Note that in the intra-deduplication, Ai judges whether the duplicate exists by comparing hash values. Thus, the search complexity is very efficient, i.e., O(1). However, in the inter-deduplication, since inter-tags are random elements in G generated by BGN encryption, the same data will correspond to different inter-tags. Thus, the CSP cannot compare the hash values of inter-tags to check the duplicate. Nevertheless, if the one by one search method is adopted, the time complexity of duplicate search increases linearly with the number of encrypted data stored in the CSP, which will lead to huge search costs, especially for big data. Accordingly, we construct a deduplication search tree (DDT) based on the B+ tree for efficiently searching duplicates.

Algorithm 1:
Search algorithm in the deduplication decision tree
Function: Search(Lm,node)
1: The search function Search(Lm,node) is started from the root node, i.e., node = root, and it proceeds to a leaf node as follows:
2: if node is a leaf then
3: return node
4: else
5: case 1: Lm < δ1 then
return Search(Lm,P0)
case 2: δk 6 Lm < δk+1 then
return Search(Lm,Pk)
case 3: δnc−1 6 Lm then return Search(Lm,Pnc−1)
9: end if

Algorithm 2:
Inter-deduplication algorithm
1: For the received data (i,Lm,τˆm), the CSP calls the function Search(Lm,node) in the DDT to search whether the value Lm has already been stored.
2: if the same value is not found then
3: return "data upload"
4: else
5: the same value Lm∗ is found, e.g., (j,τˆm∗,Bm∗,Cm∗), and then check whether i = j holds.
6: if i = j then
7: return "data upload"
8: else
9: verify whether the following equation holds
e(τˆm,gj)p = e(τˆm∗,gi)p (4)
10: if Eq. (4) holds then
11: return "duplicationklinkm∗kBm∗"
12: else
13: return "data upload"
14: end if
15: end if
16: end if
Based on the above analyses, we can obtain that if and only if mk = mt, i.e., h2(mk) = h2(mt), then

$$e(g,g)^{\alpha i+j\gamma ph2(mk)} = e(g,g)^{\alpha i+j\gamma ph2(mt)}$$

$$\Leftrightarrow e(\tau\hat{}mk,gj)p = e(\tau\hat{}mt,gi)p$$

Note that h2 : {0,1}∗ → {0,1}κ−1 is a cryptographic hash function, and the hash value satisfies h2(mk) < q for the arbitrary data mk.

4.3 Data encryption key recovery:

Upon receiving the message "duplicationklinkm∗kBm∗", Ai forwards "duplicationkBm∗" to the user U and stores

(Bm∗,linkm∗) together with Tm, i.e., (Tm,Bm∗,linkm∗), where m = m∗. Otherwise, Ai directly forwards the message "data upload" to U. After receiving the feedback from Ai, if the message is "data upload", U performs data encryption operations. If the received message is "duplicationkBm∗", U conducts key recovery operations. The details are described as follows. the data m before outsourcing as follows.

βm ∈ ZN, set Km = e(gn+1,g)βm, • Randomly choose

and generate the convergent key ckm as

.

Note that the value e(gn+1,g) can be computed as e(gi,gn+1−i).

- Generate the ciphertext of m as

Cm = SKEckm(m), where the set I = {1,2,...,n} is comprised of the identifiers of n domains and SKEckm(•) represents a fast symmetric encryption algorithm, like AES. Finally, the ciphertext tuple of the data m is (Bm,Cm).

Note that the ciphertext Bm is used to encapsulate the random value Km. The idea of computing Bm comes from the broadcast encryption scheme. After data encryption, U sends (Bm,Cm) to Ai. Upon receiving (Bm,Cm), Ai forwards it to the CSP and stores (Tm,Bm). Finally, the CSP stores (i,Lm,τˆm,Bm,Cm) in the appropriate location of the DDT by leveraging Algorithm 1, and sends the logical link linkm to Ai for storing. That is, Ai finally stores

(Tm,Bm,linkm).

Key recovery: If the message is "duplicationkBm∗", then U

recovers the convergent key with the private key di and the data m as follows.

- Let Bm∗ = (B0,B1), compute
- Then, U can recover ckm∗ with the data m as : ckm∗ =

. Note that in this case, the data m

is the duplicate of the data m∗, i.e., m = m∗.

The correctness of Eq. (7): Based on Formula (6), the Accordingly, we can obtain.

Finally, U deletes the data m and just stores the information

(Tm,ckm,labelm), where , labelm is the

label of the data m, e.g., the name or feature used for identifying the data m). The reason for using labelm is to make it easy for users to identify each data. Note that if m is the duplicate, then , where e(gn+1,g)βm∗ is generated by the first uploader. Otherwise, is generated by himself.

4.4 Data Download

After a period of time, when the user U from Di wants to download an outsourced data m, U uses the corresponding label labelm to find the stored Tm, and then sends a request "downloadkTm" to Ai. After receiving the request "downloadkTm", Ai finds the same hash value and get the corresponding link linkm. Then, Ai returns linkm to U. After that, U can directly download the ciphertext Cm from the CSP based on the linkm.
Finally, U decrypts Cm with the stored convergent key ckm
and verifies data integrity. The details are introduced as follows.

- U recovers m0 by decrypting Cm with ckm.
- Then, with the recovered m0, U computes Tm0 =

, and verifies the integrity by checking whether Tm0 = Tm holds. If it does not hold, it means that m has been corrupted, i.e., m0 6= m.
, if a user from Di has previously tried to upload the data m. After the data deduplication

data. Therefore, users only need to interact with Ai to obtain the corresponding logical link, which can reduce the download time to a great extent compared to the interaction with the CSP.

4.5 MQ135 Sensor Definition

1. (Computational Diffie-Hellman (CDH) Problem). The CDH problem in G is defined as follows: Given g,ga,gb ∈ G for unknown a,b ∈ ZN, compute gab ∈ G.
The n-BDHE problem is: given a vector of 2n+
1 elements
G2n+1 for unknown α ∈ ZN, compute e g,h)αn+1 ∈ GT. G2n+1 for unknown α ∈ ZN, compute e g,h)αn+1 ∈ GT.
It is worth noting that the input vector is missing the term gαn+1 so that the bilinear map seems to be of little help in computing the required e g,h)αn+1.
Definition 3. (Decisional n-BDHE problem). The decisional n-
BDHE problem is stated as follows: given (2n + 1) elements
and
R ∈ GT for an unknown random α ∈ ZN, determine whether R = e g,h)αn+1 or a random element from GT.
Definition 4. (The (decisional) n-BDHE assumption) We say that the (decisional) n-BDHE assumption holds if no probabilistic and polynomial-time algorithm has advantage at least in solving the (decisional) n-BDHE problem.

4.6 Boneh-Goh-Nissim Cryptosystem:

Boneh-Goh-Nissim cryptosystem (BGN) includes three algorithms: key generation, encryption, and decryption. The details are described as follows.

- Key generation: Given a security parameter κ, run Gen(κ) to get a tuple (p,q,G,GT,e) as described in Section 3.1 and set N = pq. Randomly choose two generators g,x ∈ G and set y = xq. The private key is p and the corresponding public key is pk = (N,G,GT,e,g,y).
- Encryption: Given a message m ∈ {0,1,...,W}, where
, choose a random number r ∈ ZN, and compute the ciphertext as C = gmyr ∈ G.
- Decryption: Given the ciphertext C and private key p, compute Cp = (gmyr)p = (gp)m. Let g0 = gp, then
Cp = g0m. To obtain m, it suffices to compute the discrete logarithm of g0m. Actually, when m is a short message, say The Boneh-Goh-Nissim cryptosystem (BGN) includes three algorithms: key generation, encryption, and decryption. The details are described as follows.

- Key generation: Given a security parameter κ, run Gen(κ) to get a tuple (p,q,G,GT,e) as described in Section 3.1 and set N = pq. Randomly choose two generators g,x ∈ G and set y = xq. The private key is p and the corresponding public key is pk = (N,G,GT,e,g,y).
- Encryption: Given a message m ∈ {0,1,...,W}, where
, choose a random number r ∈ ZN, and compute the ciphertext as C = gmyr ∈ G.
- Decryption: Given the ciphertext C and private key p, compute Cp = (gmyr)p = (gp)m. Let g0 = gp, then
Cp = g0m. To obtain m, it suffices to compute the discrete logarithm of g0m. Actually, when m is a short message, say m 6 W for some small bound√ W, the decryption takes expected time O(W) utilizing Pollard's lambda method [24].

4.7 B+ Tree:

The B+ tree of order f is a very efficient, dynamic and balanced search tree that satisfies the following properties:

- The root node has at most f children and at least two children if it is not a leaf node.
- Each non-leaf node with nc children contains nc − 1 keywords: (P0,δ1,P1,δ2,P2,...,δnc−1,Pnc−1), where
- δk, k = 1,2,...,nc−1, is a keyword with δk−1 < δk.
- Pk is a pointer that points to the root of the subtree. All the keywords pointed by Pk−1 are smaller than δk, but greater than or equal to δk−1.
- The number of children is satisfied .Each leaf node (unless it is a root) must contain nc − 1 keyword and pointer pairs: (δ1,P1,...,δnc−1,Pnc−1), where All data is stored in leaf nodes, and the pointer Pk, k = 1,...,nc − 1, points to the data that contains the keyword δk.

THE PROPOSED SCHEME which includes three phases: system setup, data upload, and data download.

System Setup:

The trusted KDS generates private keys for users, a secret for the CSP and the corresponding system parameters. Specifically, the KDS first runs the composite bilinear parameter generator Gen(κ0) to output a tuple (N = pq,G,GT,e). Then, the KDS generates system parameters and private keys as follows.

- Randomly choose two generators g,x ∈ G and two numbers α,γ ∈ ZN, and then compute y = xq, ν = gγ and gi = gαi for i = 1,2,...,n,n + 2,...,2n.
- For all users in Di, compute the private key , where i = 1,2,...,n. Note that n is the total number of domains in system and i is the identifier of Di.
- Choose three cryptographic hash functions: h1 : {0,1}∗ → {0,1}κ1, h2 : {0,1}∗ → {0,1}κ0−1 and h3 : G → {0,1}κ1, where κ1 is the bit length of the convergent key.
- Finally, the KDS sends di to all users in Di and p to the CSP by secure channels, and publishes system parameters pp = (N,G,GT,e,h1,h2,h3,y,g,g1,...,gn,gn+2,...,g2n,ν).

The proposed scheme can easily support the addition of users to the existing domains. Specifically, suppose that a user U∗ is added to the domain Di, we can introduce a simple identity verification protocol that allows U∗ to verify the legitimacy to the KDS. if the number of domains is 10 in the current situation, then we can set n to 20. The reserved portion can be used for subsequent domain additions.

4.9 Data Upload:

The data upload phase mainly includes four parts: *tag generation*, *intra-deduplication*, *inter-deduplication* and *data encryption / key.* For each user U in D_i, where i = 1,2,...,n, when U wants to upload the data m, U first generates an intra-tag for data deduplication. Then, the agent A_i performs the intra-deduplication to check the duplicate in the same domain D_i. If the duplicate does not exist, then the CSP needs to further conduct the interdeduplication among different domains, **.** Finally, if the duplicate is found, then U recovers the convergent key generated by the first uploader. Otherwise, U encrypts m and uploads the corresponding ciphertexts. The details are described as follows.

5. SECURITY ANALYSIS

In particular, following the design goals illustrated in Section 2.3, our analysis will focus on how our scheme can achieve data confidentiality and data integrity while resisting brute-force attacks.

5.1 Data Confidentiality

Data confidentiality of our scheme includes two aspects: (1) The semantic security of encrypted data and tags; (2) The preservation of the message equality information. The details are analyzed as follows.

5.1.1 Semantic security of encrypted data and tags

In our scheme, any adversary even corrupting the CSP and A_i or unauthorized users cannot feasibly extract any information about a plaintext from its ciphertext or tag.

First, we analyze that A_i and the CSP cannot obtain the content from the stored or received data. Specifically, A_i can receive the data (L_m,τ_m,B_m,C_m) from the user U in D_i. With the private key a_i, A_i can obtain $g^{\alpha i\gamma h2(m)}$ from the intra-tag τ_m (see Eq. (2)). If A_i wants to obtain the data m from $g^{\alpha i\gamma h2(m)}$, A_i first needs to overcome the discrete logarithm problem (DLP) to obtain $\alpha^i\gamma h_2(m)$, and then deals with the integer factorization to obtain $h_2(m)$. At last, A_i has to compute the one-way hash function to get m. However, the DLP, integer factorization and one-way hash function have been proved to be computationally infeasible difficult problems [28]. Hence, it is infeasible to get m from the intra-tag τ_m. If A_i wants to get m from (B_m,C_m), A_i tries to get the random element $K_m = e(g_{n+1},g)^{\beta m}$, which is the crux of the convergent key $ck_m = h_1(K_m\|m)$. However, based on the n-BDHE hard problem (see Definition 2), A_i cannot obtain K_m from the parameters $(g^{\beta m},g_1,...,g_n,g_{n+2},...,g_{2n})$. Besides, based on the decisional n-BDHE problem (see Definition 3), without the private key d_i, A_i cannot recover K_m from B_m (see [27] for details). Without knowing the random K_m, the convergent ciphertext C_m is a random ciphertext generated by the symmetric encryption algorithm, which satisfies the semantic security [4], [29]. Note that the value L_m denotes the size of the data m, it is a feature that every data possesses. In general, this value would not leak any information about the data content. Therefore, A_i cannot obtain any information about the data m from the obtained data(L_m,τ_m,B_m,C_m).

The CSP stores $(i,L_m,\hat{\tau}_m,B_m,C_m)$ received from A_i. Similarly, without the private key d_i, the CSP cannot obtain the data m from the ciphertext tuple (B_m,C_m). For the inter-tag $\hat{\tau}_m = g^{\alpha i\gamma h2(m)}y^r$ generated by the BGN encryption, the CSP possesses the secret p and can compute $(\hat{\tau}_m)^p = (g^p)^{\alpha i\gamma h2(m)}$.

Recall in Section 3.2, when the plaintext space is small, say $W \ll q$, it is possible to compute the discrete logarithm with Pollard's lambda method. However, the parameters α, γ and $h_2(m)$ belong to a large space. At this point, the discrete logarithm becomes a hard problem. Thus, the CSP cannot obtain any private information (i.e, the data m or the private parameters α and γ).

$$T_{m^*} = h_3(d_i^{h_2(m^*)})$$

Then, we analyze that users lacking ownership cannot read the data content from the encrypted data. For any user U in D_i, if U tries to obtain the data that does not own by himself, U can choose a random number H such that the bit length of H is κ_0-1. Then, U generates an intra-tag $\tau_H = (d_i^H g^{a_i r_H}, g^{r_H})$ with the private key d_i. After that, U interacts with A_i to execute the data upload phase. If an identical hash value happens to be stored, i.e., $H = h_2(m^*)$, then U can receive the corresponding ciphertext B_{m^*}. Accordingly, U can recover the value $K_{m^*} = e(g_{n+1},g)^{\beta m^*}$ from B_{m^*} by computing Eq. (7). However, U does not possess the data m^*, and thus U cannot generate the convergent key $ck_{m^*} = h_1(K_{m^*}\|m^*)$. In fact, the probability that a randomly selected H equals to $h_2(m^*)$ is very small, i.e., $P = \frac{1}{2^{\kappa_0-1}}$. Thus, when the security parameter κ_0 is large enough, it is almost impossible to obtain K_{m^*} through such attempts.

5.1.2 Preservation of message equality information

We prove that our scheme can protect the message equality information from disclosure to some extent. Actually, in order to achieve data deduplication, the message equality information has to be leaked. Fortunately, our scheme can minimize such disclosure. Specifically, during the intra-deduplication, before obtaining the hash value T_m to compare, Eq. (2) has to be performed, i.e., $d_i^{h_2(m)} g^{a_i r_m} / (g^{r_m})^{a_i} = d_i^{h_2(m)}$. Since only A_i possesses the private key a_i, based on the CDH problem (see Definition 1), only it can compute Eq. (2). In other words, only A_i knows the message equality information in the domain D_i. In the inter-deduplication, to verify whether two inter-tags from different domains.

5.2 Data Integrity

After obtaining the data m', it is very necessary for U to verify data integrity to ensure the stored data m' is not altered. The reason is that deduplication techniques only keep one copy. Without the assurance of data integrity, once the outsourced data is altered, U cannot obtain the correct original data and even cannot be aware of such change. Our scheme allows users to easily verify data integrity. Specifically, U generates the value $T_{m'} = h_3(d_i^{h_2(m')})$. Recall that after the data upload phase, U stores the hash value $T_m = h_3(d_i^{h_2(m)})$. Then U checks whether $T_{m'} = T_m$ holds. As we consider hash functions h_2 and h_3 as random oracles, if $T_{m'} \neq T_m$, then $m' \neq m$, which implies that the downloaded data m' is corrupted.

5.3 Brute-Force Attacks Resistance

In this section, we analyze that our scheme can prevent the CSP, A_i or the malicious user (corrupted by the adversary) from obtaining the content of the targeted ciphertext by launching bruteforce attacks.

5.3.1 Security against offline brute-force attacks

First, we consider an offline brute-force attack launched by the CSP or A_i on the accessible information. Suppose the CSP knows the background knowledge of the plaintext space M, and wishes to know which data corresponds to the specific ciphertext or tag. That is, for a stored data $(i,\hat{\tau}_m,B_m,C_m)$, where $\hat{\tau}_m = d_i^{h_2(m)} y^r$, $B_m = (g^{\beta_m}, (\nu \cdot \prod_{k\in\mathcal{I}} g_{n+1-k})^{\beta_m})$ and $C_m = \mathrm{SKE}_{ckm}(m)$, the CSP wishes to determine which data $m_t \in$ M generates them by launching offline brute-force attacks. More specifically, with the secret key p, the CSP computes $(\hat{\tau}_m)^p$ to obtain $d_i^{p\cdot h_2(m)}$. For each data $m_t \in$ M, the CSP first computes the hash value $h_2(m_t)$, and then wishes to generate $d_i^{h_2(m_t)p}$ to check whether $d_i^{p\cdot h_2(m_t)} = d_i^{p\cdot h_2(m)}$ holds. If it holds, then it implies that $h_2(m) = h_2(m_t)$, i.e., $m = m_t$. Unfortunately, the CSP cannot generate the valid value $d_i^{h_2(m_t)p}$ without the private key d_i. Thus, the CSP cannot obtain the content from the inter-tag by an offline brute-force attack. Moreover, without the private key d_i, the CSP cannot obtain the value $K_m = e(g_{n+1},g)^{\beta m}$ from B_m or public parameters due to the (decisional) n-BDHE hard problem. Accordingly, the CSP cannot generate the valid convergent key $ck_{mt} = h_1(K_m\|m_t)$ without K_m. That is, the CSP can neither generate a valid ciphertext C_{mt} nor decrypt C_m to judge whether $m_t = m$ or not. Therefore, the CSP cannot determine which plaintext corresponds to the specific encrypted data and tag by launching offline brute-force attacks. In our threat model, A_i also wants to obtain the content of the stored data by launching offline brute-force attacks. Specifically, for the obtained data (τ_m,B_m,C_m), where $\tau_m = h^2(m)^{airm}, g^{rm})$, A_i can use the private key a_i to obtain (d_i $gh^2(m)$ d_i from the intra-tag τ_m. Similarly, for each data $m_t \in$ M, A_i

computes $h_2(m_t)$ and tries to generate $d_i^{h_2(m_t)}$ to check $h^2(m^t)$ $h^2(m)$ whether $d_i = d_i$ holds. However, it cannot generate a valid $d_i^{h_2(m_t)}$ without d_i. Similar to the CSP, A_i cannot obtain useful information from the encrypted data (B_m, C_m). Therefore, A_i also cannot guess the content of the targeted ciphertext or tag by launching offline brute-force attacks.

5.3.2 Security against online brute-force attacks

In addition to the CSP and A_i, a malicious user (corrupted by the adversary) from the domain D_i may launch an online bruteforce attack by repeatedly executing the data upload procedure. For each data $m_t \in$ M, he/she generates the intra-tag τ_{mt} and tries to upload m_t to observe whether actual uploading of m_t has occurred. The online brute-force attack repeats this procedure until the duplicate exists. That is, the malicious user can know the content corresponding to the targeted ciphertext.

Although such an attack cannot be completely prevented because the user has the private key, the actual effect of the attack can be minimized. Similar to existing works associated with bruteforce attacks [5], [8], [11], we adopt the rate-limiting strategy to mitigate such attack. Specifically, we use the bounded rate-limiting approach of DupLESS [5]. In this approach, each A_i limits the total number of data upload requests a user can make during a fixed interval of time, e.g., A_i can set the maximum number of times to perform Eq. (2). By doing so, we can greatly reduce the capability of the malicious user to execute this online brute-force attack.

6. PERFORMANCE EVALUATION

We evaluate the performance of our scheme in terms of computational, communication and storage overheads. Moreover, we give a comparison with Shin et al's scheme [8] and Jiang et al's scheme [30]. Since these three schemes use a symmetric encryption algorithm to generate the data ciphertext, we ignore related overheads of the same part in the comparison.

6.1 Theoretical analysis

In this section, we discuss the theoretical analysis and give a comparison of three schemes. Similarly to [8] and [30], we ignore the costs of the KDS since it does not participate in data upload and download phases. Besides, compared to the exponentiation and pairing operations, the computational cost of a hash operation is very fast, which can be ignored. In what follows, we describe computational costs and communication overhead of uploading and downloading the data m, and also show the related storage costs.

6.1.1 Computational Cost

For the sake of simplicity, we use t_m, t_{mt}, t_e, t_{et} and t_p to represent the computational cost of a multiplication in G, a multiplication in GT, an exponentiation in G, an exponentiation in GT and a pairing, respectively.

1. Computational cost of our scheme. In the phase of data up load, an intra-tag τ_m is first generated, which requires $3t_e + t_m$. Then, A_i computes Eq. (2) and T_m to check the duplicate, which costs $t_e + t_m$. If the duplicate exists, then A_i returns "duplicationkB_m∗" to the user. Otherwise, A_i computes the inter-tag $\hat{\tau}_m$, which costs $t_e + t_m$. After that, the CSP receives $(i, L_m, \hat{\tau}_m)$ and performs the inter-deduplication. Based on Algorithm 2, the CSP first searches whether the value L_m is recorded. The search complexity in DDT is $O(\log_f |S_{all}|)$, where $|S_{all}|$ denotes the number of stored data in the CSP. If the same value is found, the CSP needs to verify whether Eq. (4) holds, which costs $2t_p$. Note that $e(\hat{\tau}_m, g_j)^p = e(\hat{\tau}_m, g_j^p)$. Once the system is set up, the value g_j^p, $j = 1, \ldots, n$, can be computed offline by the CSP and will remain unchanged. Thus, the cost of g_j^p can be ignored. After finishing data deduplication, if the duplicate is not found, the user encrypts m as (B_m, C_m). In addition to the cost of the C_m, the additional costs are $2t_e + t_{et}$. Note that, B_m is computed as

$$B_m = \left(g^{\beta_m}, (\nu \cdot \textstyle\prod_{k \in \mathcal{I}} g_{n+1-k})^{\beta_m}\right).$$

Similarly, once the system is set up by the KDS, the value $(\nu \cdot {}^{Q}{}_{k\in I}\, g_{n+1-k})$ is a constant. Accordingly, $(n-1)t_e$ can be ignored. However, if the duplicate is found, the user needs to recover the convergent key ck_m∗. The crux is to obtain the value $e(g_{n+1}, g)^{\beta m*}$ by computing Eq. (7). Likewise, for each user from D_i, $d_i \cdot {}^{Q}{}_{k} \in I, k6_{=i} g_{n+1-k+i}$ can be considered as a constant when the system is set up. Therefore, the corresponding costs are $t_{mt} + 2t_p$. In the phase of data download, it contains the decryption and data verification. Since we ignore the cost of the symmetric encryption algorithm, we only discuss data verification. With the decrypted data m^0, the user computes $T_{m'} = h_3(d_i^{h_2(m')})$, and then verifies the integrity by checking whether $T_m 0 = T_m$ holds. Thus, the cost of data download is t_e.

2. Comparison of computational cost. In order to make a comparison, we list the computational costs of three schemes in Table 1. Note that, $|S_{all}|$ and $S_{Di}|$ denote the number of encrypted data stored in the CSP and from the D_i, respectively. l is the bit length of short hash used in [8] and usually around 5 ~ 20 bits.

6.1.2 Communication Overhead

In order to facilitate the analysis, we use |G| and |GT| to denote the bit length of an element in G and GT, respectively. |L|, |$link$| and |i| denote the bit length of the data length L_m, the data link link$_m$ and the domain identifier i, respectively.

1. Communication overhead of our scheme. When a user U in D_i wants to upload data m, U first sends "uploadk(L_m,τ_m)" to A_i, which costs |L| + 2|G| bits. If the duplicate is found, then A_i returns "duplicationkB_m∗" with 2|G| bits in length to U. If the duplicate is not found, then A_i sends "uploadk(i,L_m,$\hat{\tau}_m$)" with |i| + |L| + |G| bits in length to the CSP for the further inter-deduplication. Then the CSP returns "duplicationk$link_m$∗kB_m∗" with |$link$| + 2|G| bits in length to A_i when the duplicate is found. Otherwise, the CSP returns "upload" to A_i. After the inter-deduplication, A_i either returns "duplicationkB_m∗" with 2|G| bits in length or directly forwards "upload" to U. If the received message is "upload", then U sends the encrypted data (B_m,C_m) to A_i. Except for the symmetric ciphertext C_m, the additional overhead is 2|G| bits. Then, A_i forwards (B_m,C_m) to the CSP to obtain the corresponding link link$_m$, which costs |$link$| + 2|G| bits to transmit. If the received message is "duplicationkB_m∗", U does not need to upload any encrypted data. Finally, when the duplicate is found in the intradeduplication, total overheads of data upload require |L| + 4|G| bits. If the duplicate is found in the inter-deduplication, then totaloverheads are 7|G|+2|L|+|i|+|$link$| bits. Inversely, when the duplicate is not found, total overheads are 7|G|+2|L|+|i|+|$link$| bits.

6.1.3 Storage Cost

In our scheme, after data upload, the user deletes the data m and just stores (T_m,ck_m,label$_m$), which needs 2κ_1 + |$label$| bits. After the intra-deduplication, if the duplicate exists, then A_i does not need to store any data. Otherwise, A_i records (T_m,B_m,link$_m$) with 2|G|+|$link$|+κ_1 bits in length. Similarly, if the duplication exists in the inter-deduplication, the CSP would not store any data. Otherwise, the CSP stores the outsourced data (i,L_m,$\hat{\tau}_m$,B_m,C_m). Except for the symmetric ciphertext C_m, the additional storage costs for the CSP are 3|G| + |L| + |i| bits.

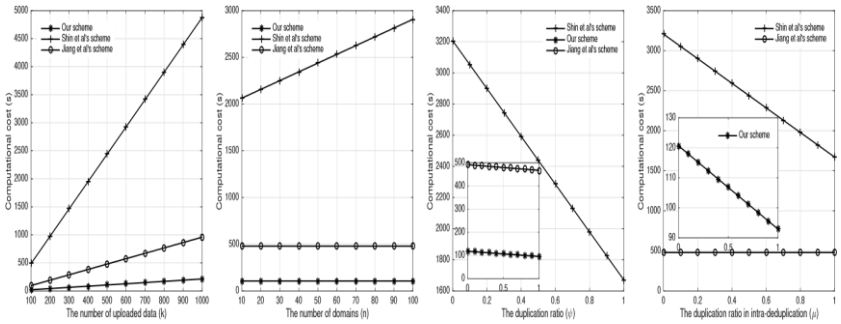
6.2 Simulation analysis

In this section, we discuss the simulation analysis and give a comparison of three schemes. Specifically, we conduct experiments with PBC [31] and OpenSSL [32] libraries running on a 2.6 GHzprocessor 2 GB-memory computing machine.

6.2.1 Simulation results

1. Computational cost. We depict the comparison of computational costs in Fig. 4. Specifically, Fig. 4(a)-4(e) show the variation of operation cost in terms of k, n, ψ, and μ. Fig. 4(f)-4(h) show the variation of the duplicate search complexity in terms of total stored data number $|S_{all}|$ and the order of the DDT f (or disjoint subset number 2^l in [8]).

From Fig. 4, we can see that the computational efficiency of our scheme is much better than the other two schemes for all variables. The main reasons are two aspects: the efficiency of ciphertext calculations and the duplicate search complexity. On the one hand, for ensuring that the outsourced data can be correctly decrypted by users with ownership, our scheme only contains a constant number of ciphertexts, i.e., a symmetric ciphertext C_m and a key-encapsulated ciphertext B_m with two elements in G. However, in the scheme [8], except for the symmetric ciphertext, the key-encapsulated ciphertexts $\{e(H(m)^s, g)r, (g^{\frac{1}{y_1}})^s, \ldots, (g^{\frac{1}{y_n}})^s\}$contain one element in G$T$ and n elements in G, which increases linearly with n.



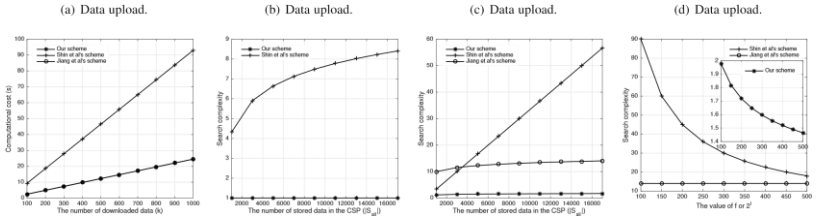
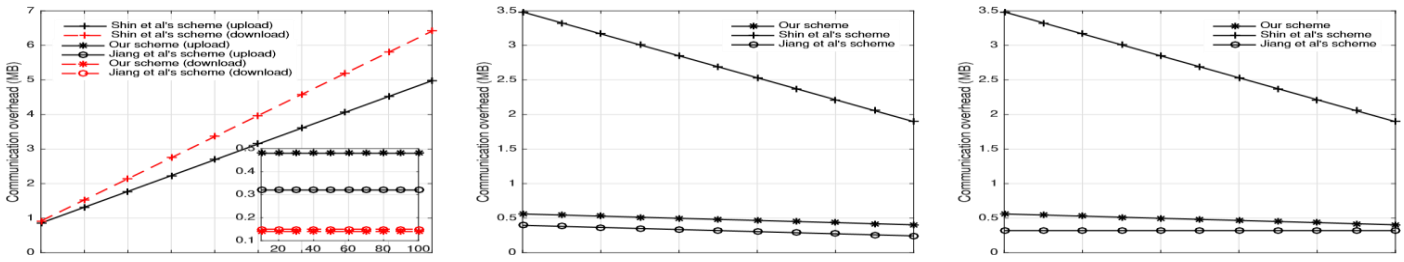


Fig 4. Notification Alerts

Thus, our scheme can significantly reduce the computational costs of data encryption, as shown in Fig. 4(a)-4(d). Note that, in the scheme [30], it directly uses the hash value of the data m as the main key to obtain the symmetric key. In other words, as long as the user has $h(m)$, the symmetric key selected by the initial uploader can be obtained correctly. Thus, this scheme would not introduce a large number of additional ciphertexts as in the scheme [8]. However, it is clear that this scheme is not resistant to brute-force attacks. On the other hand, in the intra-deduplication, our scheme constructs a hash table to find the duplicate, where the search complexity is $O(1)$. But the scheme [8] uses a binary search tree to store the tag, where the search complexity is $O(\log_2 |S_{Di}|)$. For convenience, we directly set $|S_{Di}| = |S_{all}|/n$. Obviously, the search complexity of our scheme is more efficient than the scheme [8], as shown in Fig. 4(f). For the inter-deduplication, our scheme constructs the DDT based on the B+ tree, where the search complexity is $O(\log_f |S_{all}|)$. The scheme [30] uses the binary search tree to find the duplicate, where the search complexity is $O(\log_2 |S_{all}|)$. However, the scheme [8] introduces a short hash to divide the set S_{all} into 2^l subsets, and thus the search complexity is sub-linear, i.e., $O(|S_{all}|/2^l)$. In general, the logarithmic search complexity is more efficient than the sub-linear complexity, while the complexity of the B+ tree is more efficient than the binary search tree. Therefore, the search complexity of our scheme is more efficient than the other two, as shown in Fig. 4(g) and 4(h).

For data download, the costs of our scheme and Jiang et al's scheme [30] are less than the Shin's scheme [8] for k, as shown in Fig. 4(e). The reason is that our scheme achieves data verification by computing $T_{m^*} = h_3(d_i^{h_2(m^*)})$. However, the scheme [8] needs to verify whether $e(g,t_m) = e(H(m^*),g^{xi})$ holds. Note that, the scheme [30] does not consider data verification. The resulting costs are related to the recovery of the symmetric key.

Communication overhead. In Fig. 5, we plot the comparison of communication overheads in terms of k, n, ψ, and μ. It is shown that our scheme significantly reduces communication overheads compared with the scheme [8]. Similar to the analysis of computational cost, the main reason is that our scheme only sends a constant number of ciphertexts to the CSP. However, for the scheme [8], the number of uploaded or downloaded ciphertexts is linear with n, which will cause more communication overheads, especially when n increases, as shown in Fig. 5(b). The communication overhead of our scheme is slightly larger than the scheme [30]. The reason is that we introduce the agent between users and the CSP to perform the deduplication and forward the received messages to the CSP or users, which will increase communication overheads



6.2.2. Storage cost.

Fig. 6 shows the comparison of storage costs for these three schemes in terms of k, n, ψ, and μ. It is shown that our scheme reduces storage costs compared with the scheme [8]. The main reason is that the number of ciphertexts generated in our scheme is constant. Note that for the convenience of comparison, we summarize the storage cost of A_i to the CSP for analysis. Due to the introduction of the agent, the storage cost of our scheme is slightly larger than the scheme [30], as shown in Fig. 6(b)6(d). But, this process helps greatly improve the efficiency of the duplicate search if the data come from the same domain, as shown in Fig. 4. More importantly, the agent can maintain a ratinglimiting strategy to resist the online brute-force attacks launched by malicious users. From the above analyses, our scheme is indeed efficient in terms of computational,

communication and storage overheads compared with the up-to-date works, which shows the significance and practical potential of our scheme to support big data.

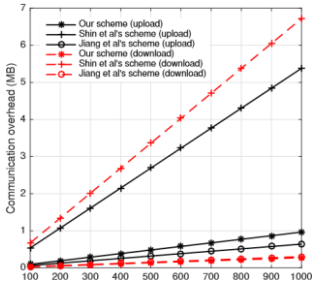


Fig. 5: The comparison of communication overhead.

7. RELATED WORK: Secure data deduplication technique, as it can eliminate redundant data while achieving data confidentiality, has been widely developed by the research community [12]. The convergent encryption (CE) was the first solution that can achieve deduplication over encrypted data [3], [33]. In the CE, data is encrypted by its hash value and the corresponding tag is produced by hashing the generated ciphertext. Clearly, when different users independently encrypt the same data, they will generate the same ciphertexts and tags which can be easily deduplicated. However, the precise security guarantee of the CE was never fully proven or even stated. After that, Bellare et al. [4] formalized the CE a new cryptographic primitive.

8. CONCLUSION AND FUTURE ENHANCEMENT :

In this paper, we have proposed an efficient and privacy-preserving big data deduplication scheme for a two-level multi-domain architecture. Specifically, our scheme can achieve the semantic security of encrypted data while ensuring that the outsourced encrypted data can be correctly decrypted by users with ownership by generating a constant number of ciphertexts. Besides, only the CSP (the agent) can decide whether two given random inter-tags from different domains (intra-tags from the same domain) correspond to the same data or not, which minimizes the disclosure of the message equality information. Detailed security analyses demonstrate that our scheme can achieve data confidentiality and data integrity while resisting brute-force attacks. Furthermore, extensive performance evaluations show that our scheme outperforms the existing competing schemes, especially the computational cost and the time complexity of the duplicate search.

Future research includes extending the proposed scheme to achieve the ownership management and revocation because data modification or deletion operations are often requested by users. Moreover, since data deduplication techniques keep only one copy of the data, outsourced data after deduplication is vulnerable to data loss or corruption. Accordingly, future research agenda will also include extending the proposed scheme to address the reliability of outsourced data.

REFERENCES

[1] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," TOS, vol. 7, no. 4, pp. 14:1–14:20, 2012.

[2] 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, 2010.

[3] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in ICDCS, 2002, pp. 617–624.

[4] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings, 2013, pp. 296–312.

[5] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: Server-aided encryption for deduplicated storage," in Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013, 2013, pp. 179–194.

[6] M. Miao, J. Wang, H. Li, and X. Chen, "Secure multi-server-aided data deduplication in cloud computing," Pervasive and Mobile Computing, vol. 24, pp. 129–137, 2015.

[7] Y. Duan, "Distributed key generation for encrypted deduplication: Achieving the strongest privacy," in Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, CCSW '14, Scottsdale, Arizona, USA, November 7, 2014, 2014, pp. 57–68.

[8] Y. Shin, D. Koo, J. Yun, and J. Hur, "Decentralized server-aided encryption for secure deduplication in cloud storage," IEEE Transactions on Services Computing, vol. PP, no. 99, pp. 1–1, 2017.