Efficient and Secure Storage for Outsourced Data: A Survey

Jianfeng Wang[1] · Xiaofeng Chen[1]

Received: 9 August 2016 / Revised: 18 August 2016 / Accepted: 22 August 2016 / Published online: 6 September 2016

Abstract With the growing popularity of cloud computing, more and more enterprises and individuals tend to store their sensitive data on the cloud in order to reduce the cost of data management. However, new security and privacy challenges arise when the data stored in the cloud due to the loss of data control by the data owner. This paper focuses on the techniques of verifiable data storage and secure data deduplication. We firstly summarize and classify the state-of-the-art research on cloud data storage mechanism. Then, we present some potential research directions for secure data outsourcing.

Keywords Outsourced storage · Verifiable search · Data auditing · Secure data deduplication

1 Introduction

Cloud computing, the new term for the long-dreamed vision of computing as a utility, can offer plenty of benefits for real-world applications, such as on-demand self-service, ubiquitous network access, rapid resource elasticity, usage-based pricing, outsourcing, etc. One of the fundamental advantages of cloud computing is the so-called outsourcing paradigm. That is, the resource-constrained users can enjoy high-quality data storage services by outsourcing their data to the cloud server.

Despite the tremendous benefits, the outsourcing paradigm brings some new security challenges. On the one hand, the cloud server may be not fully trusted, and face both internal and external security threats, such as software/hardware failures, compromised employees, hacker. A query on data stored on a cloud server may return an invalid search result. What's more, the cloud server may be "semi-honest-but-curious" and intentionally execute partial search operations in order to save its computation and communication overhead. Thus, one significant security challenge is how to achieve the *verifiability* of search results for data stored in the cloud. It means that the client should efficiently check the validation for the results returned by the cloud server. Specifically, the following two security requirements should be meet: (1) correctness: the result is the original data and has not been modified; (2) completeness: the result includes all the matched data satisfying the client's search request.

On the other hand, with the rapid popularity of cloud computing, an increasing amount of data is being outsourced to the cloud in a exponential growth manner. Inevitably, this leads to a cost explosion of data storage. This concerns not only the cost of the hardware and software necessary for storing data, but also the rapidly growing energy consumption in storage systems. As a promising solution, data deduplication has attracted increasing attention from both academic and industrial community. Deduplication can eliminate redundant data by storing one single copy for duplicate data.

In this paper, we present a comprehensive survey of solutions for verifiability of search results and secure data deduplication. Specifically, we first review the state of the art for verifiable data search and secure data deduplication techniques, and introduce a classification of these techniques. Then, we present current research directions, with the aim of promoting further research of data security in the cloud.

✉ Jianfeng Wang
wjf01@163.com

[1] State Key Laboratory of Integrated Service Networks (ISN), Xidian University, Xi'an, People's Republic of China

The rest of the paper is organized as follows. In Sect. 2, we briefly present an brief overview of verifiable cloud storage, including security threats and the corresponding solutions. A summary of secure data deduplication in cloud environments is presented in Sect. 3. Finally, we discuss some future research directions for secure data outsourcing and conclude this paper in Sect. 4.

2 Verifiable Storage on Outsourced Databases

Database outsourcing has recently attracted considerable interest. The concept of database outsourcing was first implicitly introduced by Hacigümüş et al. [22]. Their approach allows the data owner to delegate the database management to a cloud service provider (CSP) that provides various database services to users. More specifically, in the outsourced database (ODB) scenario, the data owner locally encrypts its own database and then outsources the encrypted database with additional metadata (i.e., index) to the CSP, which hosts the database and provides various database services to the users on behalf of data owner. The data users can issue query to the CSP and receive the corresponding results from the CSP.

Despite the tremendous benefits, the outsourced database paradigm inevitably suffers from some new security challenges. Specifically, due to self-interest and hardware/software failures, cloud servers may execute only a fraction of the search operations honestly and/or return an incorrect and/or incomplete query result. What is worst is that, since users no longer locally possess a copy of the data, it is difficult to check the integrity of search result. Therefore, one of the most critical challenges is to effectively audit the integrity of outsourced databases.

2.1 System Model

As shown in Fig. 1, an ODB system consists of three entities: the data owner, the data user, and the cloud service provider. The data owner outsources its encrypted database to the cloud service provider, and an (authorized) data user can issue encrypted queries to the CSP. It is worth nothing that the CSP should be able to process queries over encrypted data. In addition, the data user should be able to verify the search result. Verifiability includes the following two security goals: (1) correctness: the result is the original data and has not been modified; (2) completeness: the result includes all valid data items satisfying the search condition.

2.2 Threat Model

In an ODB system, the CSP refers to a “semi-honest-but-curious” server. That is, the CSP may not honestly follow the proposed protocol but return incomplete search result and/or execute only partial search operations honestly. Thus, two types of attacker are considered: (1) external attacker: a party which wants to obtain knowledge on the database beyond what the party is authorized to obtain, i.e., a revoked user or hacker. (2) internal attacker: a party may have some knowledge about database (i.e., the CSP). The goal of the attacker is to return incomplete/incorrect search results without being detected.

2.3 Integrity Auditing for Outsourced Databases

Several researchers have investigated techniques for verifiable database outsourcing in the past decade. The existing approaches can be categorized into two types according to the verification approach adotped.

2.3.1 Authenticated Data Structure-Based Integrity Verification

The first approach is based on authenticated data structures (e.g., Merkle hash tree [36]) [8, 16, 17, 27, 34, 38, 47]. Devanbu et al. [17] firstly investigated the problem of integrity auditing on outsourced databases. Their solution does not require the results pre-computation (signature) of all the possible queries nor deliver the whole database to the user. The basic idea is that an index based on the Merkle hash tree (MHT) is generated, and then, the search result can be verified by re-computing the signature of the root of the MHT. Note that the leaf nodes of a MHT should be ordered. Such requirement makes frequent data updates costly. More importantly, the size of verification object (VO) is linear in the cardinality of the query result and logarithmic in the scale of the database. Pang et al. [47] proposed the notion of verifiable B-trees (VB-tree), where each internal node is assigned with a signed hash value derived from all the data items in the subtree rooted at the current node. In processing a query, the cloud server first locates the smallest subtree covering all the query results. It then computes the VO as the hash values for all the data items in the subtree that are not included in the result. Trivially, the size of VO is independent of the database size. Nuckolls [45] presented a flexible verification structure called hybrid authentication tree (HAT) by incorporating one-way accumulator. The proposed solution enables a consolidated proof to reduce the size of the VO. Later, Li et al. [27] introduced a novel notion of Embedded Merkle B-tree (EMB-tree). The basic idea is to embed a B^+-tree into an MHT. To verify the completeness of a range query, the VO includes all the sibling hash out of scope of two immediately neighboring records in the ordered sequence.

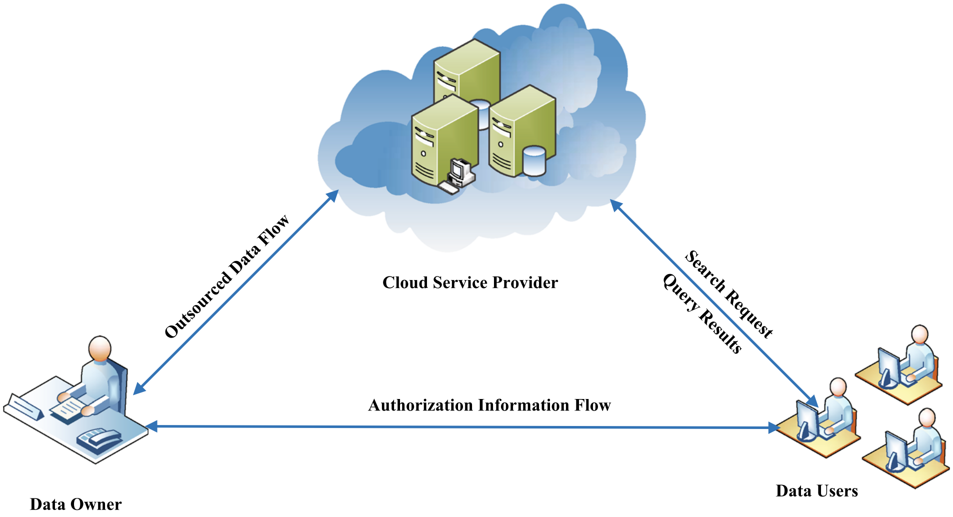

Fig. 1 Architecture of an outsourced database system model

2.3.2 Signature Chaining-Based Integrity Verification

The second approach is based on the signature chaining technique [40, 41, 43, 46]. Mykletun et al. [40] investigated the notion of signature aggregation which allows one to combine multiple signatures into a single one, thereby reducing verification overhead for search results. However, their mechanism ensures correctness for search results and does not provide completeness guarantee. Later, Narasimha et al. [43] addressed completeness by integrating signature aggregation and chaining techniques. Specifically, the client generates a signature for each data item containing all the immediate predecessors in different dimensions. Then, given a range query, their technique requires two boundary data items to be returned along with the target data items. The completeness of the search result can be verified using the chained signature. Pang et al. [46, 48] give two solutions to the completeness problem for static and dynamic outsourced database, respectively. In their solutions, all data items are assumed to be ordered with respect to certain searchable attributes, and the data owner creates a signature for each item that consists of information about the two neighboring items in the ordered sequence. Note that there is no need for additional boundary data items. Nevertheless, the case when non-continuous regions are queried is intractable. Recently, Yuan and Yu [59] presented a new verifiable aggregation query scheme for outsourced databases. Specifically, each data item is assigned an authentication tag based on a polynomial, which can be used to check the integrity of query result for certain aggregation queries.

Notice that none of the existing solutions ensure the completeness of result when the cloud server intentionally returns an empty result. Wang et al. [55] proposed a novel verifiable outsourced database scheme based on Bloom filters. In their construction, the data user can check the integrity of the search result even if the CSP intentionally returns an empty set. Their technique allows the data user to ensure the correctness of search result by checking whether the search request belongs to the Bloom filters.

Remark 1 As a complementary solution, a probabilistic integrity verification methods have been proposed by Xie at al. [57] and Sion [50]. The main idea of such methods is that the data owner inserts some faked data items in the database beforehand. The disadvantages of such methods are twofold: On the one hand, the fake data items must be shared by all authorized data users and this makes the methods vulnerable to compromise attacks. On the other hand, the methods requires the cloud server to return all attributes of the data items and thus the method cannot support some common database operations such as projection.

Remark 2 Another concern about data integrity auditing is related storage integrity for outsourced data. Storage integrity refers to the ability to check whether the outsourced data are lost or corrupted without retrieving it. The pioneer works include the Provable Data Possession (PDP) protocol [3] and the Proof of Retrievability (POR) protocol [26]. Since the definition of such protocols, several researchers have investigated the problem of remote data auditing.

It should be pointed out that there are some differences between storage and query integrity for outsourced data. First, in the storage integrity setting, the user must have beforehand knowledge about the database (e.g., the hash value of data blocks). By contrast, in the query integrity setting, the user is not required to have knowledge about the database. Furthermore, storage integrity only focuses on query correctness whereas query integrity must ensure both the correctness and completeness of the query.

2.4 Verifiable Databases with Updates

Benabbas et al. [7] proposed a useful cryptographic primitive for verifiable databases with efficient updates (VDB). That is, a resource-constrained client may outsource a large-scale database to a cloud server and later efficiently performs verification of query results in a dynamic database scenario. If a dishonest cloud server tampers with any data item in the database, the misbehavior will be detected with an overwhelming probability (once the tampered data item is queried). In addition, the cost of query processing and query result verification should be independent of the size of the database.

For the case of static databases, the above problem can be addressed by trivially adopting message authentication or digital signature technique. Namely, the client signs each data item before uploading it to the cloud server, and the cloud server is required to return the requested data item together with its valid signature. Nevertheless, this solution cannot work well when the database is updated. The main challenge is related to how to revoke the valid signatures given to the cloud server for the previous values of the modified data item. A naive solution is that the client locally keeps track of every change. However, such a solution negates the advantages of database outsourcing. Although existing techniques such as accumulators [10, 11, 44], and authentication data structures [35, 42, 49, 52] that can be adopted to address such problem, these solutions either rely on nonconstant size assumptions (e.g., the q-Strong Diffie-Hellman assumptions) or require expensive operations such as the generation of primes and re-shuffling procedures.

Benabbas et al. [7] proposed the first efficient VDB scheme under the subgroup membership assumption in composite order bilinear groups. The main idea is to apply a verifiable polynomial evaluation scheme constructed with algebraic pseudo-random functions. However, their solution can only achieve private verifiability. In other words, only the data owner can perform verification of search results. As the data users have limited resources, it is critical that any data user verify the validity of data updated by the server. Here, we introduce the formal definition of VDB [7]. In the definition, the term "client" refers to the notion of "data owner" that we use in our discussion throughout the paper.

Definition 1 A verifiable database scheme $\mathsf{VDB} = (\mathsf{Setup}, \mathsf{Query}, \mathsf{Verify}, \mathsf{Update})$ consists of four algorithms defined as follows.

- $\mathsf{Setup}(1^k, DB) \rightarrow (\mathsf{S}, \mathsf{PK}, \mathsf{SK})$: On input the security parameter k and a database DB, the setup algorithm is run by the client to generate a database encoding S that is given to the server, a public key PK that is distributed to all users, and a secret key SK that is secretly stored at the client.
- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x) \rightarrow \sigma$: The query algorithm takes as input an index x and returns a pair $\sigma = (v, \pi)$, which is run by the server.
- $\mathsf{Verify}(\mathsf{PK}/\mathsf{SK}, x, \sigma) \rightarrow v$: The public verification algorithm outputs a value v if σ is correct with respect to x, and an special symbol $\perp$ otherwise.
- $\mathsf{Update}(\mathsf{SK}, x, v'_x) \rightarrow \mathsf{PK}'$: In the update algorithm, the client firstly generates a token t'_x with its own secret key SK and then sends the pair (t'_x, v'_x) to the server. Then, the server uses v'_x to update the database record of index x, and outputs the updated public key PK' according to t'_x.

2.4.1 Vector Commitment-Based VDB Framework

Catalano and Fiore [12] formalized a powerful cryptographic primitive named vector commitment. Informally speaking, the notion of vector commitment allows one to commit to an vector $(m_1, \ldots, m_q)$ in such a way that the committer can later open the commitment at specific positions. Also, nobody should be able to open a commitment to two different values at the same position (this is called *position binding*). Besides, the vector commitment should be *concise*, i.e., the size of the commitment string and the opening are both independent of the vector commitment q. Additionally, the vector commitment should be updatable for constructing a VDB scheme. That is, it is required that the committer is able to update the original commitment value by changing a specific component of the vector and the opening would still be valid for the updated commitment. The detailed formal definition of vector commitment can be found in [12].

Catalano and Fiore [12] constructed a novel VDB scheme from the vector commitment. The proposed construction does not only rely on the standard constant-size cryptographic assumption (Computational Diffie-Hellman), but also satisfies the property of public verifiability. Formally, the framework consists of the following algorithms:

- $\mathsf{Setup}(1^k, DB) \rightarrow (\mathsf{S}, \mathsf{PK}, \mathsf{SK}$: Let the database be $DB = (i, v_i)$ for $1 \leq i \leq q$. Run the key generation and committing algorithms of vector commitment to obtain the public parameters $\mathsf{PP} \leftarrow \mathsf{VC.KeyGen}(1^k, q)$ and the initial commitment and auxiliary information $(C, \mathsf{aux}) \leftarrow \mathsf{VC.Com}_{PP}(v_1, \cdots, v_q)$, respectively. It outputs the database encoding $\mathsf{S} = (\mathsf{PP}, \mathsf{aux}, DB)$, the system public key $\mathsf{PK} = (\mathsf{PP}, C)$ and the client's secret key $\mathsf{SK} = \perp$.

- $\mathsf{Query}(\mathsf{PK}, \mathsf{S}, x) \rightarrow \sigma$: On input an index x, the server firstly runs the opening algorithm to compute $\pi_x \leftarrow \mathsf{VC.Open}_{PP}(v_x, x, \mathsf{aux})$ and then returns $\sigma = (v_x, \pi_x)$.
- $\mathsf{Verify}(\mathsf{PK}, x, \sigma) \rightarrow v_x$: Parse the proofs σ as (v_x, π_x). If $\mathsf{VC.Ver}_{PP}(C, x, v_x, \pi_x) = 1$, then return v_x, and an special symbol $\perp$ otherwise.
- $\mathsf{Update}(\mathsf{SK}, x, v'_x)$: To update the record of index x, the client firstly retrieves the current record v_x from the server. That is, the client obtains $\sigma \leftarrow \mathsf{Query}(\mathsf{PK}, \mathsf{S}, x)$ from the server and checks that $\mathsf{Verify}(\mathsf{PK}, x, \sigma) = v_x \neq \perp$. Then the client computes $(C', U) \leftarrow \mathsf{VC}.\mathit{Update}_{PP}(C, v_x, x, v'_x)$ and outputs $\mathsf{PK}' = (\mathsf{PP}, C')$ and $t'_x = (\mathsf{PK}', v'_x, U)$. Finally, the server uses v'_x to update the database record of index x, PK' to update the public key, and U to update the auxiliary information.

2.4.2 *Weaknesses of Catalano–Fiore's VDB Scheme*

Chen et al. [14] described two types of attack for the Catalano–Fiore's VDB scheme, namely the Forward Automatic Update (FAU) attack and the Backward Substitution Update (BSU) attack. We revisit them in what follows:

2.4.2.1 Forward Automatic Update (FAU) Attack In the Catalano–Fiore's VDB scheme [12], anyone (include a malicious cloud server) can update the data in the same manner of the data owner. To be specific, an adversary first retrieves a record v_x. Then, the adversary generates the new public key PK' and the token t' based on a new data record value v' (without involving any knowledge of secret key). Finally, the cloud server updates the corresponding data record as well as the public key. Interestingly, any query issued to the cloud server can be replied to with a valid proof based on the forward updated public key PK'. As a consequence, the above misbehavior would not be detected. The result is that an auditor cannot determine with certainty that the cloud server has been dishonest.

2.4.2.2 Backward Substitution Update (BSU) Attack The so-called BSU attack means that anyone can substitute the current public key with the previous one. As noted above, anyone is allowed to update the public key. Therefore, if the client does not locally store a copy of the public key; it is difficult for him to distinguish the past public key from the latest one. On the other hand, even if the client has stored the latest public key, it is still to be difficult for the client to prove that the locally stored public key is the latest one.

Remark 3 The main reason of the above attacks is that the client's secret key is not be involved in the update of the public key. Note that it is useless to append the signature on the public key. If the cloud server generates the signature, it has the ability to compute the signature on any public key. On the other hand, if the signature is computed by the client, the original question arises again: How to efficiently revoke the previous (valid) signature?

2.4.3 *VDB Framework from Commitment Binding*

To achieve public verifiability and protect against the FAU/BSU attacks simultaneously, Chen et al. [14] proposed a novel VDB framework for vector commitment based on the idea of commitment binding (see Fig. 2). That is, the client uses the secret key to generate a signature on some binding information. This information consists of the latest public key, the commitment on the current database, and a global counter. Assume that the client's signature on the binding information is $H_T = \mathrm{SIGN}_{sk}(C_{T-1}, C^{(T)}, T)$, then we obtain the current public key as $C_T = H_T C^{(T)}$. Thus, this method recursively binds the commitment C_T to a 3-tuple $(C_{T-1}, C^{(T)}, T)$. As a consequence, an adversary (i.e., the cloud server) cannot update the database and public key without the client's secret key.

Chen et al. [15] also introduced the notion of verifiable database with incremental updates (Inc-VDB), i.e., the client can efficiently update the ciphertext with the previous one, rather than from scratch. It is useful for large database settings, especially when there are frequent slight modifications. Note that in traditional encryption schemes the ciphertext needs to be totally recomputed even if only one single bit is changed in plaintext , resulting in very high overhead for the resources-constrained clients. To address this challenge, the Inc-VDB framework incorporates vector commitment and encrypt-then-incremental MAC mode of encryption [9]. The main trick is that the updated ciphertext v'_x is generated in an incremental manner as follows: we define $v'_x = (v_x, P_x)$, where $P_x = (p_1, p_2, \ldots, p_k)$ denotes the location of bit positions with different values between the original and updated plaintext message. Given $v' = (v_x, P_x)$, the client first decrypts v_x to

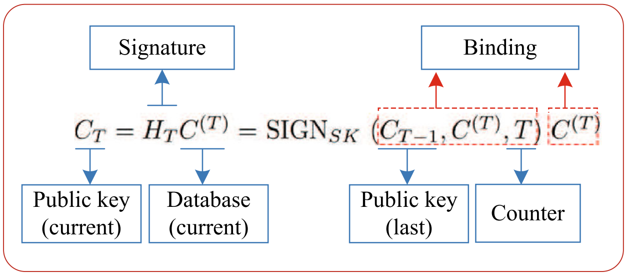

Fig. 2 Commitment binding technique

obtain m_x and then performs the bit flipping operation on the positions of P_x to obtain the final plaintext m_x'.

It worth noting that the existing VDB schemes cannot fully support data update operations. Specifically, the existing solutions can only support data replacement and deletion operations while they are not applicable of the insertion operation. The main reason is that the number of index of the database must be fixed in advance and published as the system parameters in both schemes. On the other hand, when the client performs an insertion/deletion operation on an outsourced database, the number of the index will be increased/ decreased by 1. Therefore, it seems to be a paradox to design a VDB scheme that supports all update operations using the existing solutions, such as delegating high-degree polynomial function and vector commitment.

Miao et al. [37] utilized the idea of hierarchical (vector) commitment to address the above dilemma. The hierarchical commitment consists of multiple levels, and the maximum number of data items for each level is the dimension q of vector in a vector commitment. When a level is a full (i.e., the number of data items in this level is q), a new inserted data record will be located in a new level.

3 Secure Cloud Data Deduplication Technique

According to the latest analysis by IDC [54], the volume of data we create and copy annually is doubling in size every 2 years, and will reach 44 trillion gigabytes in 2020. With the dramatic increase in data volumes, how to efficiently store the ever-increasing data becomes a critical challenge for cloud servers. Data deduplication, as a specialized data compression technique, has been adopted widely to save storage costs by only storing a single copy of repeating data and replacing with links to that copy. Data deduplication can achieve more than 50 % storage reduction [2] and has been deployed by many cloud storage providers, such as Dropbox, Google Drive, Bitcasa and Mozy.

However, conventional encryption is incompatible with deduplication. Specifically, encrypting the same data with different encryption keys results into distinct ciphertexts corresponding to the same source data. Thus, it makes cross-user deduplication impossible.

To fill the above gap, convergent encryption (CE) [19], an elegant cryptographic primitive, is proposed. Essentially speaking, CE is a deterministic symmetric encryption scheme and its encryption key is derived from the cryptographic hash value of the file content. Then, each identical data item generates the same ciphertext, which achieves deduplication and encryption simultaneously. Bellare et al. [6] defined a new cryptographic primitive called message-locked encryption (MLE), which can be viewed as a generalization of CE. Furthermore, to enhance performance of deduplication, a randomized convergent encryption (RCE) scheme has been proposed. It is characterized by the efficiency of the relevant operations, i.e., key generation, message encryption, and tag production. However, RCE is vulnerable to what is called duplicate faking attack. Specifically, an honest user cannot retrieve his original message because it can be undetectably replaced by a fake one. To tackle this problem, an interactive version of RCE, called interactive randomized convergent encryption (IRCE) [4], has been proposed. In IRCE, an honest user can check tag consistency by interacting with the server and thus verify that the original ciphertext is stored. If an adversary may upload a modified ciphertext, this ciphertext will be inconsistent with respect to the corresponding file tag. Such a mismatch allows one to detect that the ciphertext is incorrect.

3.1 Deduplication Classification

According to the granularity and architecture, deduplication can be categorized into different types. With respect to granularity, there are two deduplication strategies. (1) File-level deduplication: The data redundancy is exploited at the file level. Only one copy of the identical data file is saved and subsequent copies are replaced with a link that points to the original file. (2) Block-level deduplication: each file is divided into multiple blocks (or segments, chunks) and the data redundancy is exploited at the block level. Note that the block size can be either fixed or variable in practice. Despite achieving higher deduplication ratio, block-level deduplication inevitably requires more metadata and needs longer processing times.

With respect to the architecture, there are two deduplication strategies. (1) Server-side deduplication (known as target-based deduplication): All clients upload their data to the CSP and are unaware of deduplication that might occur. The CSP is responsible for deleting the duplicate copies. This strategy reduces storage costs, but does not save bandwidth costs. (2) Client-side deduplication (known as source-based deduplication): The client first sends a tag of the data (e.g., a hash value) to the CSP to check whether the data to be uploaded are already in the cloud. If yes, the data does not need to be uploaded. This strategy can save both bandwidth and storage costs, but is prone to side channel attacks since a client can learn if another client already uploaded a given file. The details will be discussed later.

3.2 Security Challenges and Solutions

Without loss of generality, we focus on client-side, cross-user deduplication. Cross-user deduplication means that the deduplication operations are performed across all data uploaded by all users. Such method increases the effectiveness of deduplication, as deduplication is executed not only when a single user repeatedly uploads the same data but also when different users upload the same data.

Despite its benefits in reducing storage and communication costs, client-side cross-user deduplication suffers from several privacy threats [5, 24, 58].

3.2.1 Brute-Force Attack

As discuss above, CE protocols can be used to ensure data privacy in deduplication. However, it is vulnerable to brute-force attacks. That is, suppose the target message is drawn from a finite space of size n $S = \{M_1, \ldots, M_n\}$. Then, any attacker can generate the convergent key of each message and compute the corresponding ciphertext as in off-line encryption. If one computed ciphertext is equal to the target ciphertext, the target message is inferred. The basic reason is that CE is a deterministic symmetric encryption scheme, and the key space is limited. It implies that no MLE (CE) scheme can achieve traditional semantic security [21]. The ideal security for MLE scheme, PRV$-CDA [6], refers to an encryption scheme that can achieve semantic security when the messages are unpredictable (i.e., have high min-entropy).

Bellare et al. [5] proposed a novel secure deduplication system resisting brute-force attacks, called DupLESS, which can transform the predictable message into an unpredictable one with the help of an additional key server. More specifically, DupLESS introduces an additional key server that generates the convergent key based on two inputs: the hash of message and a system-wide key. The client obtains the convergent key by interactively running an oblivious pseudorandom function (OPRF) with the key server. As long as the key server is secure, the convergent key is derived from a random large key space. It implies that DupLESS can ensure confidentiality for the predictable message. Furthermore, to prevent online brute-force attacks by compromised client, a per-client rate-limiting strategy is applied to limit the total number of queries a client can make during each epoch. It implies that DupLESS can achieve the same security of MLE at worst even the key server is compromised. Duan [20] proposed a distributed version of DupLESS, where the client must interact with the threshold of other clients to generate the convergent key before uploading a file. Moreover, a trusted dealer should be included to distribute key shares for each client. We argue that the trusted dealer has similar role of the key server in DupLESS. Thus, this scheme still suffers from online brute-force attacks in the case in which the dealer is comprised.

Recently, Liu et al. [33] proposed a secure single-server cross-user deduplication scheme that resists brute-force attacks. The client who wants to upload a given file runs a password authenticated key exchange (PAKE) protocol with the CSP to obtain the encryption key from the original client who had previously uploaded the identical file. Suppose the client wants to upload a file, the client first sends a short hash of the uploading file as "password." Upon receiving the uploading request, the CSP firstly identifies all the candidate clients with the same short hash value and asks the client to engage in a Same-Input-PAKE protocol with each candidate client. Note that as the Same-Input-PAKE protocol is run between the CSP and the client, the client does not need direct communications among themselves. To protect against brute-force attacks, two additional mechanisms are introduced. First, it uses the randomized threshold strategy [24] to assign a random threshold for each file and perform client-side deduplication once the number of the file is higher than the threshold value, so that the attacker cannot determine whether the file being uploaded already exists at the CSP. Second, a per-file rate-limiting strategy is used to protect against online brute-force attacks. Compared with the per-client rate-limiting strategy in DupLESS, the proposed strategy enhances security of deduplication and reduces communication overhead (i.e., the run time of PAKE).

3.2.2 Duplicate Faking Attacks

In a duplicate faking attack, an honest user might be unable to retrieve his original file, since it can be replaced by a fake one and the replacement cannot be detected. That is, suppose that users Alice and Bob possess two different files F_a and F_b, respectively. The malicious user Alice may upload a modified ciphertext $C_a = (E(H(F_b), F_a))$ and the corresponding tag $T_a = H(E(H(F_b), F_b))$ into the CSP. Later, when the honest user Bob uploads the ciphertext $C_b = (E(H(F_b), F_b))$ and its tag $T_b = H(E(H(F_b), F_b))$, the CSP wrongly determines that the plaintexts of C_b and C_a are identical, and thus deletes C_b. As a result, Bob cannot retrieve his original plaintext. The main reason is that the CSP cannot check tag consistency [6] without knowing the hash value of the file.

To address this drawback, a variant of MLE called randomized convergent encryption (RCE) has been introduced [6]. RCE introduces a checking mechanism, called guarded decryption, by which the client can check the integrity of the returned ciphertext. The RCE scheme is described as follows: the client first picks at random a key L and then computes ciphertext $C_1 = E(L, F)$ and $C_2 = L \oplus K$, where K is the hash value of file $H(F)$. The tag is generated from the file by a double hash, i.e., $T = H(K)$. Upon receiving the ciphertexts C_1 and C_2, the tag T, the client can obtain the random key $L = C_2 \oplus K$ using the hash of file, and the plaintext F by decrypting C_1 with L. Then, the client regenerates a tag $T' = H(H(F))$ and checks whether T' is equal to T. Furthermore, an interactive version of RCE, called interactive randomized convergent encryption (IRCE), has been proposed [4], by which a client can check the consistency of a file tag by interacting with the CSP. In this way, the client can ensure that the original ciphertext is stored by the CSP. However, the cloud server cannot check consistency between the tag and the ciphertext, since it has no access to the original plaintext. Thus, the CSP cannot determine which user is dishonest. From the point of view of practical applications, this is a major drawback. Preferably, it should be possible not only to identify which users are malicious, but also to trace these users—i.e., identify all ciphertexts uploaded by these. This is nontrivial, if a CSP or the data owners allow users to remain anonymous or appear under different identities. Wang et al. [56] designed a novel deduplication scheme, called TrDup, which makes it possible to trace malicious users. Specifically, each user generates a kind of anonymous signature for the uploaded file—a variant of the traceable signature scheme is used. Once a duplicate faking attack is detected, the tracing agent can determine the identity of the malicious user without revealing identities of other users or linking their files in the cloud.

3.2.3 Hash Manipulation Attack

Harnik et al. [24] pointed out that client-side deduplication is vulnerable to side channel attacks. That is, whenever receiving an upload request, the CSP will tell whether the uploading file has already been stored. However, an attacker may abuse the information to launch a brute-force attack by trying all possible variants of the same file. Mulazzani et al. [39] show how to carry out this attack against mainstream cloud storage provider (i.e., Dropbox). Furthermore, Halevi et al. [23] argued that an attacker can obtain the ownership of a file that he actually does not own by providing the hash of file. The main reason is that the CSP determines whether a client owns a specific file using a small piece of information about the file (i.e., hash value). Thus, anyone who possess the short hash value for a specific file can be allowed to access the entire content of file.

To protect against such attack, Halevi et al. [23] introduced the concept of *proof of ownership* (PoW), which can be used to ensure data privacy and confidentiality in case of client-side deduplication. Namely, a user can efficiently prove to the cloud storage server that he indeed owns a file without uploading it. Three concrete PoW constructions have been presented—all based on a Merkle hash tree (MHT) built from the content of a data file.

Specifically, a challenge/response protocol is run between server and client. Each data file is denoted as a MHT (the leaf nodes constitute the data file), and the server first asks for a random subset of the MHT leaf nodes from the client. If the client does not possess the whole file, it cannot generate a valid proof with overwhelming probability. Using a PoW, the cheating attacks can be prevented. That is, a user that only knows only the hash signature of a file cannot convince the cloud server that he owns that file. Di Pietro and Sorniotti [18] proposed an efficient PoW scheme, in which each challenge is a seed for a pseudorandom generator and the response is the set of values in the file at bit positions derived by the generator from the seed. Every time a file is uploaded to the server, the latter computes a set of challenges for that file and stores them for a later check. Alís et al. [1] proposed a PoW scheme based on a Bloom filter, which is efficient at both the server and the client side.

3.3 Deduplication Efficiency

Recent approaches to secure deduplication have focused on security enhancements and efficiency. [2, 13, 25, 28–32, 51, 53, 60, 61]. Among those works, most are focused on security enhancement and efficiency improvement. Stanek et al. [51] proposed a novel deduplication encryption scheme that can provide different security levels for data files according their *popularity* that refers to how frequently the file is shared among users. Their approach can achieve a fine-grained trade-off between the storage efficiency and data security for the outsourced data. Armknecht et al. [2] designed a novel verifiable deduplication storage system, namely ClearBox, which ensures that the client can check the deduplication pattern of his own encrypted data, i.e., whether his files are deduplicated or not. Li et al. [31] and Hur et al. [25] have investigated the key update and user revocation problems, respectively.

Li et al. [28] have proposed DeKey [29], an efficient and reliable key management scheme for block-level deduplication. In DeKey, each client distributes the convergent key shares across multiple servers based on the ramp secret sharing scheme. Zhou et al. [61] proposed a more fine-grained key management scheme called SecDup, which mitigates the key generation overhead by exploiting hybrid deduplication policies. Li et al. [30] proposed a fine-grained deduplication mechanism based on user privileges. A client can perform a duplication check only for the files marked with matching privileges. Li et al. [28] designed a distributed reliable deduplication scheme, which can achieve data reliability and secure deduplication simultaneously by dispersing the data shares across multiple cloud servers. Chen et al. [13] proposed a novel storage-efficient deduplication scheme, called block-level message-locked encryption (BL-MLE), in which the block keys are encapsulated into the block tag to reduce metadata storage space.

4 Conclusion and Future Work

Secure data outsourcing is an important research topic in cloud computing. Even though secure data outsourcing has been widely investigated, more research work is needed. Relevant research directions include the following:

- *Publicly Verifiable ODB* The existing ODB schemes just support private verifiability. That is, as only the data owner can check the validity of his own data because only the data owner knows the secret key. The data owner must be involved in every verification[1]. Thus, how to design a publicly verifiable ODB scheme is an interesting problem.
- *Privacy-preserving VDB* The traditional VDB schemes do not consider the privacy of users. Specifically, information about update patterns (i.e., the updated data items and the update frequency) is leaked to the CSP. A valuable research direction is how to construct a construct privacy-preserving VDB scheme.
- *User-Revokable deduplication* Although the traceability of malicious users can be achieved in secure data deduplication, the problem of user revocation still needs to be addressed in multi-user scenarios. Thus, one valuable research topic is the development of data deduplication mechanism supporting user revocation.

Acknowledgments We are grateful to the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Natural Science Foundation of China (No. 61572382), China 111 Project (No. B16037), Doctoral Fund of Ministry of Education of China (No. 20130203110004).

References

1. Alís JB, Di Pietro R, Orfila A, Sorniotti A (2014) A tunable proof of ownership scheme for deduplication using bloom filters. In: IEEE Conference on Communications and Network Security, CNS'14, pp 481–489
2. Armknecht F, Bohli J, Karame GO, Youssef F (2015) Transparent data deduplication in the cloud. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, CCS'15, pp 886–900
3. Ateniese G, Burns RC, Curtmola R, Herring J, Kissner L, Peterson ZNJ, Song DX (2007) Provable data possession at untrusted stores. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS'07, pp 598–609
4. Bellare M, Keelveedhi S (2015) Interactive message-locked encryption and secure deduplication. In: Proceedings of the 18th IACR International Conference on Practice and Theory in Public-Key Cryptography-PKC 2015, LNCS, vol 9020. Springer, pp 516–538
5. Bellare M, Keelveedhi S, Ristenpart T (2013a) Dupless: Server-aided encryption for deduplicated storage. In: Proceedings of the 22th USENIX Security Symposium, pp 179–194
6. Bellare M, Keelveedhi S, Ristenpart T (2013b) Message-locked encryption and secure deduplication. In: Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT'13, LNCS, vol 7881. Springer, pp 296–312
7. Benabbas S, Gennaro R, Vahlis Y (2011) Verifiable delegation of computation over large datasets. In: Proceedings of the 31st Annual Cryptology Conference on Advances in Cryptology, CRYPTO'11, Springer, pp 111–131
8. Bertino E, Carminati B, Ferrari E, Thuraisingham BM, Gupta A (2004) Selective and authentic third-party distribution of XML documents. IEEE Trans Knowl Data Eng 16(10):1263–1278
9. Buonanno E, Katz J, Yung M (2001) Incremental unforgeable encryption. In: Fast Software Encryption, 8th International Workshop, FSE 2001 Yokohama, Japan, April 2–4, 2001, Revised Papers, Springer, pp 109–124
10. Camenisch J, Kohlweiss M, Soriente C (2009) An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography, PKC'09, Springer, pp 481–500
11. Camenisch J, Lysyanskaya A (2002) Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, CRYPTO'02, Springer, pp 61–76
12. Catalano D, Fiore D (2013) Vector commitments and their applications. In: Proceedings of 16th International Conference on Practice and Theory in Public-Key Cryptography, PKC'13, Springer, pp 55–72
13. Chen R, Mu Y, Yang G, Guo F (2015) BL-MLE: block-level message-locked encryption for secure large file deduplication. IEEE Trans Inf Forensics Secur 10(12):2643–2652
14. Chen X, Li J, Huang X, Ma J, Lou W (2015) New publicly verifiable databases with efficient updates. IEEE Trans Dependable Secure Comput 12(5):546–556
15. Chen X, Li J, Weng J, Ma J, Lou W (2016) Verifiable computation over large database with incremental updates. IEEE Trans Comput. doi:10.1109/TC.2015.2512870
16. Devanbu PT, Gertz M, Martel CU, Stubblebine SG (2000) Authentic third-party data publication. In: Proceeddings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, pp 101–112
17. Devanbu PT, Gertz M, Martel CU, Stubblebine SG (2003) Authentic data publication over the internet. J Comput Secur 11(3):291–314
18. Di Pietro R, Sorniotti A (2012) Boosting efficiency and security in proof of ownership for deduplication. In: Proceedings of the 7th ACM Symposium on Information, Compuer and Communications Security, ASIACCS'12, pp 81–82

[1] Here, we do not distinguish between the data owner and the authorized user because they are shared the secret key for verification.

19. Douceur JR, Adya A, Bolosky WJ, Simon D, Theimer M (2002) Reclaiming space from duplicate files in a serverless distributed file system. In: Proceedings of The 22nd International Conference on Distributed Computing Systems, ICDCS'02, pp 617–624
20. Duan Y (2014) 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, pp 57–68
21. Goldwasser S, Micali S (1984) Probabilistic encryption. J Comput Syst Sci 28(2):270–299
22. Hacigümüs H, Mehrotra S, Iyer BR (2002) Providing database as a service. In: Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26–March 1, 2002, pp 29–38
23. Halevi S, Harnik D, Pinkas B, Shulman-Peleg A (2011) Proofs of ownership in remote storage systems. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11, pp 491–500
24. Harnik D, Pinkas B, Shulman-Peleg A (2010) Side channels in cloud services: deduplication in cloud storage. IEEE Secur Priv 8(6):40–47
25. Hur J, Koo D, Shin Y, Kang K (2016) Secure data deduplication with dynamic ownership management in cloud storage. IEEE Trans Knowl Data Eng. doi:10.1109/TKDE.2016.2580139
26. Juels A, Kaliski Jr BS (2007) Pors: proofs of retrievability for large files. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, VA, USA, October 28–31, 2007, pp 584–597
27. Li F, Hadjieleftheriou M, Kollios G, Reyzin L (2006) Dynamic authenticated index structures for outsourced databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'06, pp 121–132
28. Li J, Chen X, Huang X, Tang S, Xiang Y, Hassan MM, Alelaiwi A (2015) Secure distributed deduplication systems with improved reliability. IEEE Trans Comput 64(12):3569–3579
29. Li J, Chen X, Li M, Li J, Lee PPC, Lou W (2014) Secure deduplication with efficient and reliable convergent key management. IEEE Trans Parallel Distrib Syst 25(6):1615–1625
30. Li J, Li YK, Chen X, Lee PPC, Lou W (2015) A hybrid cloud approach for secure authorized deduplication. IEEE Trans Parallel Distrib Syst 26(5):1206–1216
31. Li J, Qin C, Lee PP (2016) Rekeying for encrypted deduplication storage. In: Proceedings of the 46th IEEE/IFIP International Conference on Dependable Systems and Networks, DSN'16
32. Li M, Qin C, Li J, Lee PPC (2016) Cdstore: toward reliable, secure, and cost-efficient cloud storage via convergent dispersal. IEEE Internet Comput 20(3):45–53
33. Liu J, Asokan N, Pinkas B (2015) Secure deduplication of encrypted data without additional independent servers. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12–16, 2015, pp 874–885
34. Ma D, Deng RH, Pang H, Zhou J (2005) Authenticating query results in data publishing. In: Proceedings of the 7th International Conference on Information and Communications Security, Springer, ICICS'05, pp 376–388
35. Martel CU, Nuckolls G, Devanbu PT, Gertz M, Kwong A, Stubblebine SG (2004) A general model for authenticated data structures. Algorithmica 39(1):21–41
36. Merkle RC (1980) Protocols for public key cryptosystems. In: Proceedings of the 1980 IEEE Symposium on Security and Privacy, S&P'1980, pp 122–134
37. Miao M, Wang J, Ma J (2015) New publicly verifiable databases supporting insertion operation. In: Proceedings of the 18th International Conference on Network-Based Information Systems, NBis'15, pp 640–642
38. Mouratidis K, Sacharidis D, Pang H (2009) Partially materialized digest scheme: an efficient verification method for outsourced databases. VLDB J 18(1):363–381
39. Mulazzani M, Schrittwieser S, Leithner M, Huber M, Weippl ER (2011) Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In: Proceedings of the 20th USENIX Security Symposium
40. Mykletun E, Narasimha M, Tsudik G (2004a) Authentication and integrity in outsourced databases. In: Proceedings of the Network and Distributed System Security Symposium, NDSS'04, The Internet Society
41. Mykletun E, Narasimha M, Tsudik G (2004b) Signature bouquets: immutability for aggregated/condensed signatures. In: Proceedings of the 9th European Symposium on Research Computer Security, ESORICS'04, Springer, pp 160–176
42. Naor M, Nissim K (2000) Certificate revocation and certificate update. IEEE J Sel Areas Commun 18(4):561–570
43. Narasimha M, Tsudik G (2005) DSAC: integrity for outsourced databases with signature aggregation and chaining. In: Proceedings of the 2005 ACM International Conference on Information and Knowledge Management, CIKM'05, pp 235–236
44. Nguyen L (2005) Accumulators from bilinear pairings and applications. In: Proceedings of The Cryptographers' Track at the RSA Conference, CT-RSA'05, Springer, pp 275–292
45. Nuckolls G (2005) Verified query results from hybrid authentication trees. In: Proceedings of the 19th Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy, DBSec'05, Springer, pp 84–98
46. Pang H, Jain A, Ramamritham K, Tan K (2005) Verifying completeness of relational query results in data publishing. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'05, pp 407–418
47. Pang H, Tan K (2004) Authenticating query results in edge computing. In: Proceedings of the 20th International Conference on Data Engineering, ICDE'04, pp 560–571
48. Pang H, Zhang J, Mouratidis K (2009) Scalable verification for outsourced dynamic databases. PVLDB 2(1):802–813
49. Papamanthou C, Tamassia R (2007) Time and space efficient algorithms for two-party authenticated data structures. In: Proceedings of the 9th International Conference on Information and Communications Security, ICICS'07, Springer, pp 1–15
50. Sion R (2005) Query execution assurance for outsourced databases. In: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB'05, pp 601–612
51. Stanek J, Sorniotti A, Androulaki E, Kencl L (2014) A secure data deduplication scheme for cloud storage. In: Proceedings of the 18th International Conference Financial Cryptography and Data Security, FC'14, Springer, pp 99–118
52. Tamassia R, Triandopoulos N (2010) Certification and authentication of data structures. In: Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management
53. Tang H, Cui Y, Guan C, Wu J, Weng J, Ren K (2016) Enabling ciphertext deduplication for secure cloud storage and access control. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS'16, pp 59–70
54. Turner V, Gantz J, Reinsel D, Minton S (2014) The digital universe of opportunities: rich data and the increasing value of the internet of things. IDC White Paper, April 2014
55. Wang J, Chen X, Huang X, You I, Xiang Y (2015) Verifiable auditing for outsourced database in cloud computing. IEEE Trans Comput 64(11):3293–3303
56. Wang J, Chen X, Li J, Kluczniak K, Kutylowski M (2015) A new secure data deduplication approach supporting user traceability. In: 10th International Conference on Broadband and Wireless Computing, Communication and Applications, BWCCA'15, pp 120–124

57. Xie M, Wang H, Yin J, Meng X (2007) Integrity auditing of outsourced data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB'07, pp 782–793
58. Xu J, Chang E, Zhou J (2013) Weak leakage-resilient client-side deduplication of encrypted data in cloud storage. In: 8th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '13, pp 195–206
59. Yuan J, Yu S (2013) Flexible and publicly verifiable aggregation query for outsourced databases in cloud. In: IEEE Conference on Communications and Network Security, CNS'13, pp 520–524
60. Zheng Y, Yuan X, Wang X, Jiang J, Wang C, Gui X (2015) Enabling encrypted cloud media center with secure deduplication. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14–17, 2015, pp 63–72
61. Zhou Y, Feng D, Xia W, Fu M, Huang F, Zhang Y, Li C (2015) Secdep: a user-aware efficient fine-grained secure deduplication scheme with multi-level key management. In: IEEE 31st Symposium on Mass Storage Systems and Technologies, MSST'15, pp 1–14