As the cloud computing technology develops during the last decade, outsourcing data to cloud service for storage becomes an attractive trend, which benefits in sparing efforts on heavy data maintenance and management. Nevertheless, since

the outsourced cloud storage is not fully trustworthy, it raises security concerns on how to realize data deduplication in cloud while achieving integrity auditing.

In this work, we study the problem of integrity auditing and secure deduplication on cloud data. Specifically, aiming at achieving both data integrity and deduplication in cloud, we propose two secure systems, namely SecCloud and SecCloud+. SecCloud introduces an auditing entity with a maintenance of a

MapReduce cloud, which helps clients generate data tags before uploading as well as audit the integrity of data having been stored in cloud. Compared with previous work, the computation by user in SecCloud is greatly reduced during the file uploading and auditing phases. SecCloud+ is designed motivated by the fact that

customers always want to encrypt their data before uploading, and enables integrity auditing and secure deduplication on encrypted data

Cloud storage is a model of networked enterprise storagewhere data is stored in virtualized pools of storage which are generally hosted by third parties. Cloud storage provides customers with benefits, ranging from cost saving and simplified

convenience, to mobility opportunities and scalable service.These great features attract more and more customers to utilize and storage their personal data to the cloud storage: according to the analysis report, the volume of data in cloud is expected to achieve 40 trillion gigabytes in 2020. Even though cloud storage system has been widely adopted, it fails to accommodate some important emerging needs such

as the abilities of auditing integrity of cloud files by cloud clients and detecting duplicated files by cloud servers. We illustrate both problems below.

The first problem is integrity auditing. The cloud server is able to relieve clients from the heavy burden of storage management and maintenance. The most difference of cloud storage from traditional in-house storage is that the data is

transferred via Internet and stored in an uncertain domain, not under control of the clients at all, which inevitably raises clients great concerns on the integrity of their data. These concerns originate from the fact that the cloud storage is

susceptible to security threats from both outside and inside of the cloud [1], and the uncontrolled cloud servers may passively hide some data loss incidents from the clients to maintain their reputation. What is more serious is that for saving money and space, the cloud servers might even actively and deliberately discard rarely accessed data files belonging to an ordinary client. Considering the large size of the outsourced data files and the clients’ constrained resource capabilities, the

first problem is generalized as *how can the client efficiently perform periodical integrity verifications even without the local*

*copy of data files.* The second problem is secure deduplication. The rapid

adoption of cloud services is accompanied by increasing volumes of data stored at remote cloud servers. Among these remote stored files, most of them are duplicated: according to a recent survey by EMC [2], 75% of recent digital data is

duplicated copies. This fact raises a technology namely deduplication, in which the cloud servers would like to deduplicate by keeping only a single copy for each file (or block) and make a link to the file (or block) for every client who owns

or asks to store the same file (or block). Unfortunately, this action of deduplication would lead to a number of threats potentially affecting the storage system [3][2], for example, a server telling a client that it (i.e., the client) does not need

to send the file reveals that some other client has the exact

same file, which could be sensitive sometimes. These attacks originate from the reason that the proof that the client owns a given file (or block of data) is solely based on a static, short value (in most cases the hash of the file) [3]. Thus, the second problem is generalized as *how can the cloud servers efficiently* *confirm that the client (with a certain degree assurance) owns* *the uploaded file (or block) before creating a link to this file* *(or block) for him/her.*