Secure Storage System and Key Technologies

Jiwu Shu, Zhirong Shen, Wei Xue, Yingxun Fu
Department of Computer Science and Technology
Tsinghua University
{shujw,xuewei}@tsinghua.edu.cn, {czr10,fu-yx10}mails.tsinghua.edu.cn

Abstract— With the rapid development of cloud storage, data security in storage receives great attention and becomes the top concern to block the spread development of cloud service. In this paper, we systematically study the security researches in the storage systems. We first present the design criteria that are used to evaluate a secure storage system and summarize the widely adopted key technologies. Then, we further investigate the security research in cloud storage and conclude the new challenges in the cloud environment. Finally, we give a detailed comparison among the selected secure storage systems and draw the relationship between the key technologies and the design criteria.

I. Introduction

With the rapid development of information technology, the human has witnessed the data explosion in recent years. According to a survey [39] issued by IDC, tenfold growth of digital universe are achieved over the last five years (from 2006 to 2011). The increased size of data not only causes expensive storage overhead, but also introduces complicated data management, incurring tremendous burden to the users. Based on the considerations above, more and more users choose to migrate their data to the remote storage servers, in order to avoid the facility purchase and the troublesome data management. However, since the data are out of user's physical control, a set of security challenges arises, arousing the users' worries about whether their data are safe for the following reasons. First, the user's privacy is under the threat of malicious adversary, since the sensitive data (e.g. personal email, financial data, health recode) may be learned or altered without the data owner's permission. Second, the user's data may be at the risk of being unavailable due to the unexpected accidents or mistaken operations, which will seriously affect the user's business (e.g. the online data operations). Thus, data security against unprivileged access and data availability are the two emergency problems.

To secure data in the remote storage servers, secure storage systems [15][9][6][8][11][38][16][40][43] have been well studied for the last two decades, during which various key technologies have been proposed to effectively solve the different security problems, promoting the evolution of secure storage. However, to fully understand this field, track the footprint that storage security has passed through is an important step to conclude its development and forecast the future tendency. Although some essential reviews (e.g. [16],[38]) have been published to present the challenges and research statuses of secure storage at that time, a lot of new advances have been proposed since then. Especially when the applications of cloud storage are rapidly developing recently, some security problems (e.g. data audit) neglected in the previous studies are exposed and drawing great attention, leading to the inapplicability between the previous technologies and the new problems. Thus, a timely review for the secure storage is intensively needed to give a comprehensive understanding about its recent development. This paper gives a detailed introduction about the key technologies in the existing secure storage systems and presents some extensively considered applications that are derived recently. In particular, we summarize our contributions as follows:

- We introduce the common design criteria of secure storage, summarize the key technologies that are widely used in the previous secure storage systems, and give a detailed comparison between several typical secure storage systems.
- We present the extended applications in the cloud environment and discuss the corresponding technologies.

The rest of this paper is organized as follows. Section II will introduce the common design criteria to evaluate current secure storage systems. Then we will summarize the key technologies in section III and provide some new developed security service in section IV. Finally, a comparison between some typical secure storage systems will be presented in section V and a conclusion will be given in section VI

II. The Design Criteria of Secure Storage

The secure storage aims to protect the data in the storage systems from malicious attacks or abuses. In fact, the information security has three basic references, i.e., confidentiality (C), integrity (I), and availability (A), which are widely known as the principle of "CIA". As secure storage systems have different design concerns under various real-world scenarios, few systems (e.g. Venus[41]) have taken enough efforts to achieve all the three metrics. Instead, they focus on what their scenarios really concern.

Confidentiality means that the data information should be kept secret against the unauthorized access. Among the designs in various secure storage systems, data encryption is a widely used approach to achieve data confidentiality, where the data either are encrypted before storing on the media, or are kept in plaintext form in storage and encrypted before transferring through the network channel, separating most of the existing secure storage systems into two classes, i.e., encrypt-on-disk systems and encrypt-on-wire systems. In addition, the encryption operation also draws out some related key technologies,

978-1-4673-3030-5/13/$31.00 ©2013 IEEE

such as the access control to the ciphertext, the privilege management of users, and secret key distribution, which will be discussed in section III later.

Integrity in this paper generally refers to two aspects, i.e., unpermitted modification prevention and unpermitted modification detection [16]. The former aspect indicates that no one can alter the data except the authorized users, while the latter one denotes the unpermitted alteration should be timely detected to stop the users from being misleaded by the incorrect data. The above considerations suggest users to check the integrity of the data first before accessing the file content, to ensure that the data to access are correct. In secure storage systems, the unpermitted alteration to the data are assumed to happen when data are either rested on the disk or transferred through the network, thus it is strongly suggested that the access regulation (e.g. access control mechanism) be effectively executed to prevent the unpermitted modification. To timely detect whether the data are illegally broken, the recalculation and maintenance of a compressed value (e.g. cryptographic hash value and digital signature) for every checking granularity (e.g. file or file block) are demanded.

Availability indicates that an authorized user can execute a data operation within an acceptable period of time. Actually, the researches in the field of secure storage do not care how to make the failure servers be available, but focus on how to supply the data service continually even a failure happens. The requirement leads to the development of data redundancy technologies to decrease the probability of data unavailability. However, different methods to produce data redundancy usually incurs different storage costs, thus it is still a significant work to assure the data availability while maintaining a acceptable storage overhead.

Furthermore, secure storage also has the following concerns, such as the performance and the trust relationship of the secure storage systems. Since many computational intensive operations (e.g. data encryption and cryptographic hash calculation) are introduced to provide security protection, the system performance will be inevitably downgraded if the key technologies to secure data in the storage are adopted; therefore how to balance the security and performance remains a complicated problem. In addition, according to the various scenarios with different trust models, how to establish the trust relations among the entities in the system and how to shape their responsibilities require further exploration.

III. Basic Technologies

In this section, we summarize some basic technologies that are widely used in secure storage systems to meet the design criteria above.

A. Authentication

Authentication is the first guard of secure storage system, which is to ensure the authenticity of user's identity. Existing authentication schemes in secure storage systems are generally sorted into the following categories:

- Public key infrastructure-based authentication. In secure storage systems, PKI is the most traditional authentication method. In PKI system, every user needs a certificate or public-private key pair to check the identity of each other. For examples, Corslet [14] uses certificates to authenticate between *trust domain server* (TDS) and users; Plutus[9] allows users to employ 1024 bit RSA public-private key pair to authenticate each other.
- User ID and authentication key pairs-based authentication. This way usually needs a trusted third-party to generate IDs and authentication keys. For instance, each user in CRUST [8] has its own pair which contains userID and a individual authentication key generated by a trust agency to authenticate his identity.

B. Secret Key Distribution

As we mention before, data files are usually stored in the encrypted form to resist the illegal access. In the scenario that data files are shared among multiple users, both the encrypted file content and the secret keys are required to distribute to authorized users. Therefore, an efficient and scalable distribution mechanism for the secret keys is important to the whole system.

Existing key distribution schemes can be generally classified into storage servers-dominated schemes, data owner-dominated schemes, and trusted third party-dominated schemes. In the storage servers-dominated scheme, it usually requires the data owners and the data users to place full trust on the storage servers, which is inadequate to some threat models where the storage servers are assumed to be untrusted[14] or "honest-but-curious"[1][4]. In the data owner-dominated scheme, although it reduces the trust level of storage servers, the data owners are forced to be always online to process the user's access request and distribute the corresponding secret keys, incurring considerable management overhead; for example, the file owners and the file users share a long-term key in the design of CRUST[8] to distribute the file specific keys; in Plutus[9], the data owners distribute the relevant keys to data users via an out-of-band channel. In the trusted third party-dominated scheme, it not only avoids putting complete trust on the storage servers, but also saves data owners from cumbersome management, however, it needs a fully trusted party, which is assumed to be unaware of the ciphertext; for instance, SiRiUS[6] employs the existing key distribution infrastructure (e.g. PGP public key server or IBE master key server) to involve in the key distribution and Corslet[14] introduces the TDSs for each trusted domain to verify user's access permission and return the appropriate keys.

C. Integrity Checking

When data are stored in the unreliable storage or transferred through the untrusted network channel, they are in danger of being altered or broken due to the accident operations and the attacks launched by the malicious adversaries. These potential risks draw out a challenging problem, i.e., how to timely detect the unauthorized change of the data. A straightforward way is to calculate the hash value for the whole file (e.g. SiRiUS[6] calculates a hash value of each data file), but even the access to part of the file demands the whole file download

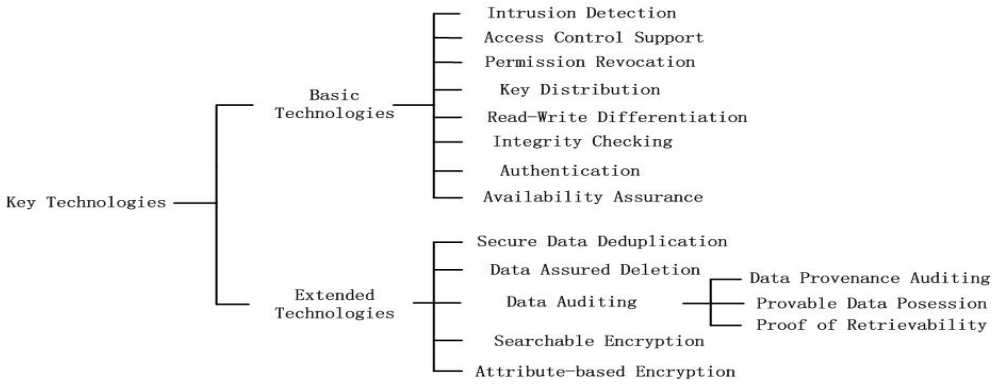

Fig. 1.: The technologies summarization introduced in this paper.

and re-computation of the hash value, which not only causes expensive cost but also leads to poor experience of random access operation. Aim to address these drawbacks, an alternative way is to partition the files into many file blocks with constant size and construct a data structure (generally, hash tree) based on the integrity information of the file blocks. During the construction of the hash tree, the leaf nodes store hash values of file blocks, while the internal nodes store hash values of the concatenation of their children, thus the root node guarantees the whole tree's integrity. Therefore, either file update or integrity checking needs to re-calculate the hash values of the nodes on the path from the associated nodes to the root node and compare them with the pre-computed hash values stored in the corresponding nodes. For example, Plutus[9] and FARSITE[15] organize the integrity information and file block keys into a Merkle Hash Tree, in which the leaf nodes store the hash values of associated file blocks and the corresponding file block keys while the root is signed by a private key; CRUST[8] organizes a hash tree, where the leaf node stores the hash value of a single file block, and the internal node has *m* children and keeps the hash value computed from the concatenation of its children.

Nowadays, the increasing amount of data that are migrated to the cloud storage triggers the appearance of another research branch of integrity checking, i.e., how to efficiently and securely perform the remote data integrity checking, where the security design is quite different. (See section IV-C for more detail).

D. Availability Assurance

As users' data are stored on the remote storage servers, users are forced to give up the physical control of their data, handing over the availability assurance to the storage servers. However, data stored in the storage servers will suffer from various kinds of Byzantine failures, putting their availability at the edge of risk. Therefore, promise the data availability is an extremely essential research to secure them in the storage.

Data redundancy across multiple servers is the most widely used technology to assure data availability. These technologies can be classified into *replication-based solution* (RBS), *erasure code-based solution* (ECBS), and *regenerating code-based solution* (RCBS). RBSs demand to duplicate the data and store the copies across different servers, where the data can be successfully retrieved as long as one copy of data survives. The possession of a file copy over each storage server makes RBSs support concurrent read/write operations very well, but they require *t* times of storage capacity to tolerant (*t*-1) failures, incurring huge storage cost. In ECBSs, each file is first partitioned into *k* file blocks, which will be further encoded into *n* coded blocks by employing a *(n,k)*-erasure code scheme. Finally, the coded blocks are distributed to different servers. The original data can be successfully recovered as long as no less than k coded blocks can be retrieved. Compared to RBSs, ECBSs greatly reduce the storage overhead while promising the data availability. However, one main drawback of ECBSs is that the high repair traffic is required for repairing an altered block. To reduce the communication cost in the repair process, RCBSs are proposed to balance the storage overhead and repair traffic, in which users can configure the storage cost and network overhead according to their needs. Generally, less repair traffic demands more storage overhead in RCBSs. The performance comparison of these three solutions is presented in Table I.

TABLE I: A comparison between RBS, ECBS, and RCBS.

Solution	Storage Requirement	Repair Traffic
RBS[a]	$t\mathcal{M}$	$\mathcal{M}$
ECBS	$\frac{n}{k}\mathcal{M}$	$\mathcal{M}$
RCBS[b]	$\frac{(2n-2)}{k(2n-k-1)}\mathcal{M}$	$\frac{(2n-2)}{k(2n-k-1)}\mathcal{M}$

[a] suppose the number of copies is *t*

[b] suppose the adopted code is the *minimal bandwidth regenerating*(MBR) code, where *k* is the number of connected healthy servers.

E. Random Access Support

Random access is an important operation, as well as a critical metrics to evaluate the performance of secure storage systems, since the whole-file read/write operation easily leads to expensive computation cost (e.g. file-level decryption and encryption) and network overhead (e.g. file-level transmission) especially when the file is large. To support random access, a general method is to set the encryption granularity to be file block-level, so users can fetch the least file blocks that cover the access range. For example, SiRiUS[6] and CRUST[8] represent each file with a series of file blocks, use the block to be the granularity of encryption and integrity checking, and

compute a signature of the root of hash tree to avoid swapping attacks.

F. Access Control

Considering the users assigned with different access permissions in the storage systems, it is essential to differentiate and limit the operations within the allowable range of each user's permission. In general the existing methods that are frequently used fall into the following classes:

- Public/private key pair-based access control. This way usually gives writers the private key and delivers the public key to readers, thus writers are authorized to append a signature to the updated files using the private key, while the readers only perform the verification to check whether the signature is legal . For example, readers (resp. writers) in Plutus[9] employ file-verify key (resp. file-sign key) to verify (resp. sign) the cryptographic hashes of file blocks, and reader in SiRiUS[6] is delivered with the public key of FSK to verify the correctness of signature while the writer is allowed to produce the signature using the private key of FSK.

- Secret keys and cryptographic hash-based access control. This way is similar to the previous one, except that the writers are permitted to produce the keys that are assigned to the readers, which significantly reduces the amount of keys that should be delivered. For example, the readers in CRUST[8] are only allowed to obtain the *"file reader MAC key"* to verify the signature of file blocks, while the writers are assigned with a *"file writer MAC key"* to sign the file blocks and given the cryptographic hash function to derive the file reader MAC key.

- Symmetric keys-based access control. This method usually originates from the concern that asymmetric cryptographic operation is far more expensive than symmetric cryptographic operation. Compared with the readers, the writers are granted extra key (keys) to append a signature. For example, Corslet[14] gives the reader a symmetric key LBK to decrypt the encrypted file blocks and grants the writer with LBK and another symmetric key FSK to update the file blocks and sign the root of Merkle Hash Tree.

- *Access control list* (ACL) and the user's public-private keys-based access control. This approach alleviates the burden of secret key distribution and cuts down the amount of keys that are required to manage at the client machines. For example, FARSITE [15] uses the authorized reader's public key to encrypt the file key that is used to encrypt the file, and stores an ACL of the writer's public keys in the metadata.

G. Permission Revocation

As a frequent operation especially in the scenario where the massive users are granted with different permissions, permission revocation happens when the file owner decides to change the permission of a certain user to the file, such as when a user quits a group or when a user behaves dishonestly. In the case that data files are stored on the trusted storage servers, the revocation only needs to delete the revoked user [15].

However, things are different when the files are hosted on the untrusted storage servers where the metrics of efficiency and security need to be taken into account. Trade-off between the two metrics derives two widely used revocation mechanisms, i.e., *aggressive revocation mechanism* and *lazy revocation mechanism*. Aggressive revocation mechanism [6] usually demands to re-encrypt the involved files with the new secret keys and re-distribute the new secret keys to the survival users once the revocation happens. Although it can timely prevent the revoked users from accessing the file content, it may cause the computation burst due to immediate re-encryption of all involved files, resulting in the instability of the whole system.

On the contrary, lazy revocation mechanism is proposed to alleviate the sudden re-encryption of the whole file and defer the re-encryption of the file blocks to the first time when the file blocks are updated after the revocation. In lazy revocation, only the secret information, whose size is far smaller than that of file blocks, should be renewed at once when the revocation is executed. The rule of lazy revocation mechanism is that it only keeps the revoked users from accessing the updated content, indicating that the revoked users can still read the unchanged file content even after the revocation is completed. At first glance, lazy revocation improves the system performance by partially sacrificing the system security, however, its adherents claim that the revoked users in the aggressive revocation mechanism can also recover the original file information, either by reading them from the cache or by copying them before the revocation, indicating that the security of lazy revocation will be not lost when compared to that of its competitor. Furthermore, lazy revocation generally needs hierarchical key management and it makes things worse when files are classified into many file groups where the files in each group originally share the same secret key, since the multi-version keys management in a file group will be activated when the revocations occur. To alleviate the problem, Plutus[9] proposes key rotation mechanism to establish the relation between the newest version of the key and the previous versions, so that a user can derive the previous keys from the current key he possesses.

H. Storage-based Intrusion Detection System

Storage-based intrusion detection system (SIDS) is a significant part of intrusion detection field to prevent unauthorized data access by analyzing access patterns and properties in storage device layer. SIDS has the following features.

First, intrusion behavior is usually accompanied with some illegal data operations. Therefore, SIDS can easily detect intrusion attack [20] by analyzing the behavior of data operations.

Second, because SIDS is generally independent of other devices without being interrupted by the intruder, thus it is workable even if the host is invaded [21][22].

Because of these characteristics, SIDS has been widely used in secure storage systems. Molina et al [22] present an independent auditor which is attached in PCI bus to check whether the system has been intruded. Based on data mining, Li et al [23] propose a new scheme, which extracts the relativity of data blocks from applications to implement this function.

IV. Extended Technologies

With the fast developing of cloud storage, the increasing amount of data is concentrated on the cloud leads to new demands for cloud security. In this section, we select some attractive technologies that are developed recently and give a systematical introduction.

A. Searchable Encryption

Data encryption is frequently used before uploading the data files to provide data security. Although this approach effectively refuses unauthorized accesses, new problems are activated especially when the amount of data rapidly increased, i.e., how to efficiently and securely retrieve the files that satisfy certain conditions. Two straightforward methods are either to download all the files and perform keyword search over plaintext, or to send the secret keys to the cloud server and let it perform keyword search after decryption. Unfortunately, these approaches either incur huge network bandwidth and computation cost, or demand users to give up the security protection. To address these problems, the designs of *searchable encryption* (SE) are proposed and receive great attention recently.

The general approach to perform keyword search over encrypted data obeys the following steps. 1) The data owner preextracts the representative keywords according to the file contents and pre-builds a searchable index before uploading the encrypted data files to the storage servers. 2) Whenever a data user wishes to access the files, he sends the search query to the data owner to apply for the corresponding search trapdoor, which will be issued to the storage servers later. 3) With the search trapdoor and the keyword taken as input, the storage servers perform the pre-design functions and send the satisfied files back to the user.

From the aspect of the constructed algorithms, the existing SE schemes can be divided into symmetric key-based SE schemes and public key-based SE schemes. In the former branch, the pseudorandom functions and cryptographic hash functions are employed, while in the latter branch, bilinear pairing is the most frequently used primitive and the security of SE is mapped to the complexity of a hard problem.

In addition, the development of SE can be classified into following categories, i.e., the works that aims to support flexible search query in the field of cryptography, and the works to support the application scenarios in the field of applied cryptography. On one hand, the flexible query significantly has an effect on the search efficiency and user's experience in SE schemes, which expresses user's search favor accurately, observably reduces the network bandwidth, and remarkably saves the efforts to process the search outputs. The support of search query is generally partitioned into three different types, i.e., single keyword query[1], conjunctive search query[45], and multi-dimensional search query with range, subset, and equality search over every dimension[44]. On the other hand, the support for different applications mainly focuses on the various application scenarios, including ranked keyword search[1], multi-keyword ranked search[4], fuzzy keyword search[2], etc. In these applications, the storage servers (or the cloud servers) are requested to learn as little information as possible and some sensitive information derived from the application should be kept secret, for example, the cloud servers should not infer the actual relevance score of each file in secure ranked keyword search[1], but only the ranked order.

With the fast growing of data stored in the cloud, the designs of SE schemes for different application scenarios remain a hotspot in near future.

B. Attribute-based Encryption

To enforce access control to the data stored on the remote storage servers, a generally adopted way in traditional secure storage systems is to encrypt the data files with certain encryption primitive and distribute the keys only to the authorized users. This method effectively prevents the unprivileged users (including the storage servers) from decrypting the ciphertext, but results in the consequence that the amount of keys grows linearly with the number of data files, which makes key management an expensive task in the scenarios with massive data files and greatly limits the scalability of the system. *Attribute-based Encryption* (ABE) [12][13] is an encryption primitive to realize expressive types of encrypted access control mechanism, where each ciphertext is tied with a pre-designed access structure and each user's private key is specified by a set of attributes according to his identity. To regulate the access to the remote data files, data owners specify a set of access structures and encrypt the data file under the corresponding structure. Only when the attributes satisfy the structure can the user obtain the secret keys to decrypt the ciphertext, thus users in the design of ABE only need to manage the attribute keys. This design significantly decreases the amount of keys that required to be kept at the client machines and allows the owners to design flexible access formulas (e.g. conjunctive/disjunctive normal form). The main research in ABE can be sorted into the following categories: the design of ABE schemes (e.g. CP-ABE [12], KP-ABE [13]), the revocation design of ABE schemes [17], the accountability of ABE schemes [19], and the design of multi-authority ABE schemes [18]. For example, Cryptographic Cloud Storage [11] introduces the application of ABE in cloud storage service to establish the security guarantee based on cryptography, rather than legal protection and manual control.

C. Data Auditing

Given that storage servers may conceal the accidents of data damage to preserve their business reputation and the unauthorized users may try to steal or tamper user's data, storage servers and unauthorized users are assumed to be untrusted in some threat models of secure storage systems. For the data owners, they are unaware of the status of their data before accessing them, leading to the delayed knowledge of their data status. To efficiently and securely audit the data status, various data auditing methods are proposed, which can be generally classified into the following branches.

Data Provenance Auditing : As the information which helps to describe the derivation history of data [24], provenance auditing is widely believed to be an important part in data auditing. For instance, Muniswamy-Reddy et al [25] make the case that provenance is crucial for cloud storage system, examine

current cloud offerings, and design three protocols for maintaining data provenance to audit the origin of data. In all, Data Provenance is a crucial tool in secure cloud storage to analyze data history.

Provable Data Possession(PDP): Nowadays, with the dramatic amount of data migrated in the storage servers, verification of the data authenticity has emerged as a critical service. PDP mechanism[26] allows users to challenge the remote storage servers to prove their possession of the data without access the entire file. To respond to the challenges, remote servers need to access part of the exact data and generate a proof to demonstrate the possession of the raw data. Current PDP mechanisms have three research branches. 1) The research to improve the efficiency of typical PDP mechanism. For example, Lou et al [27] propose a batch audit mechanism to decrease the computation overhead and network traffic. 2) The research to support data dynamic update over PDP mechanism. For example, Erway et al [29] propose D-PDP mechanism, which allows data dynamic update based on PDP mechanism. 3) The research to check whether data file has stored in servers with multiple replications. For example, Curemola et al [30] propose the MR-PDP scheme that enable users to check the possession of every replica effectively.

Proof of Retrievability(POR): PDP allows clients to check remote data's integrity, but lacks the retrievability once the data are broken. In contrast, POR scheme [31], which is composed of data possession checking mechanism and data retrieval mechanism, allows client to conditionally retrieve the broken data. The data possession checking of POR is similar as that in PDP, while the retrieval mechanism usually accomplishes by adopting the technologies of availability assurance (i.e. replica-based scheme, erasure coding-based scheme, and network coding-based scheme).

D. Data Assured Deletion

As we mention before, various kinds of schemes are proposed in many secure storage systems to strengthen data reliability, which however, also brings a potential risk once the user determines to delete his data permanently. Thus the most likely outcome is the storage servers still possess many versions (even the current version) of this data due to the effect of availability assurance technologies, which far deviates from the user's expectation. For this purpose, it needs to consider how to achieve data assured deletion.

The first data assured deletion is proposed by Perlman [32], which introduces the definition of assured deletion and designs a prototype system called ephemerizer to reach the effect once the life cycle expires. Tang et al [33] propose FADE to support a generalized policy-based assured deletion, which allows the user to delete the data that are formulated by complicated logic. Different from the above schemes that use a trusted third-party to provide assure deletion service, Geambasu et al [34] implement a proof-of-concept prototype called Vanish based on untrusted third-party (e.g. P2P). In Vanish, data will be encryptd with a key, which is then divided into k shares, encoded into n shares by using a *(n,k)*-erasure code scheme, and distributed to random DHT nodes. According to the feature of DHT, it is nearly impossible for anyone to obtain k shares to reconstruct the key when a certain cycle (usually 8 hours) elapses. Wolchok et al [35] quest about Vanish's premise, and use low-cost Sybil attack to defeat it.

E. Secure Data Deduplication

Cloud storage exposes a strong demand of deduplication, as large amount of plaintext before encryption may be duplicated. However, security protocols make deduplication in secure storage systems much more challenging for the following reasons.

First, encryption of file data makes deduplication more difficult. Since data files are encrypted into distinct ciphertext using different keys, resulting in that two identical plaintexts will usually produce almost different ciphertets with overwhelming probability. Thus traditional deduplication methods which are designed for plaintext are ineffective on ciphertext, leading to a low probability of deduplication. Due to the above reasons, rarely paper about this point appears in recent years, except one using convergence encryption[36].

Second, the access control protocol holds back the data deduplication due to data leakage concerns. For example, in the access protocol, users are asked to upload the file hash to storage servers first to check whether the file has already been kept in the servers, if it already exists, then the servers will not upload the file actually and believe the user possesses the file; thus the users can easily cheat the servers to download a target file as long as he knows the corresponding hash. Mulazzani et al [37] officially propose this problem in "online slack space".

V. Comparison and Analysis

In this section, we give a comparison between several typical secure storage systems in Table II and draw the relationship between the key technologies and the designed goals of secure storage systems in Figure 2.

Table II shows the comparison on key technologies of several typical systems. Plutus [9] is a secure storage system that aims to provide secure file sharing without placing much trust over the file servers, in which the file owner executes the operations of access control and key distribution. CRUST [8] is a stackable file system layer to provide secure file sharing over untrusted storage system and supports random access and lazy revocation. SUNDR [43] is designed to store data on untrusted servers, which supports unpermitted modification detection and places many attention to achieve fork consistency. As a platform that provides data sharing over untrusted SSP, SHAROES [40] tries to provide rich data sharing semantics and reduce the user involvement during setup and key management. Tahoe [42] is a system that supplies secure service for distributed storage. It uses cryptographic tools to achieve the protection of confidentiality and integrity, and employs erasure codes to improve data availability. Venus [41] is a service which aims for the data integrity and consistency over the untrusted cloud storage. Different from the above works, FARSITE [15] aims to provide reliable data storage. Corslet [14] is a stackable storage system that aims to provide secure file storing and sharing over existing file systems without any modification.

In Figure 2, we sort the key technologies referred in section III into four branches, which are usually employed to serve

TABLE II: The comparison between several typical secure storage systems.

Systems	Authentication	Key Distribution	Integrity Checking	Random Access	Access Control	Revocation
CRUST[8]	Use userID and individual keys to authenticate between users	Owner uses metedata to distribute the keys	Hash Tree	Yes	Distribute keys to reader/ writer via the lockboxes	Lazy revocation
Plutus[9]	Use Public-private key pairs between users via secure channel	Owner gives keys to users in out-of-band channel	Merkle Hash Tree	Yes	Use read-write keys to differentiate reader and writer	Lazy revocation
SUNDR[43]	Use public-private key pair	User exchanges public keys with superuser	File Block Hash	Yes	NA	NA
FARSITE[15]	Use user, machine and namespace signing certificates	Owner encrypts file-key and stores them with files	Merkle Hash Tree	Yes	Use ACL and public -private keys of users	Issue CA the revoked user
Tahoe[42]	NA	Owner stores encrypted keys on the server.	Merkle Hash Tree	Yes	Use the capability access control model	NA
Venus[41]	Use signature public key and email address	NA	Hash Tree	NA	Directly use commodity storage interfaces	NA
SHAROES[40]	Use public-private key pairs	Owner uses metadata to distribute keys in-band	NA	Yes	Use a cryptographic access control primitive	Aggressive revocation
Corslet[14]	Use certificates to authenticate between users and TDS	TDS distributes keys that are stored in metedata	Merkle Hash Tree	Yes	Use ACL, and two symmetric keys LBK and FSK	Lazy revocation

[a]NA here generally indicates that the technology is not referred in the system.

as the design criteria for evaluating a secure storage system. Access control, authentication, key distribution, intrusion detection and permission revocation are contained in the field of confidentiality protection, since any invalidation of these technologies will lead to data leakage to the unauthorized users. For example, if the permission revocation is carried out incorrectly, then the revoked user can still read the updated information after the revocation, which violates confidentiality protection. Based on the two metrics in integrity (i.e., unpermitted modification prevention and unpermitted modification detection), the field of integrity protection is composed of four technologies, i.e., access control, key distribution, intrusion detection, and integrity checking. For example, if the access control mechanism fails, then the reader may be allowed to modify the data files, which should be actually forbidden in the pre-designed protocol and violates the principle of unpermitted modification prevention. The integrity checking is to timely detect the unauthorized modification, obeying the principle of unpermitted modification detection. Besides, the technologies of permission revocation and random access constitute the field of performance, based on the following considerations that the support of random access can significantly reduce the overhead of read/write operations when compared to the file-level access, and the metric of performance is usually considered during the design of permission revocation which promotes the emergency of lazy revocation. Finally, the technology to ensure data availability is assigned to the field of availability assurance.

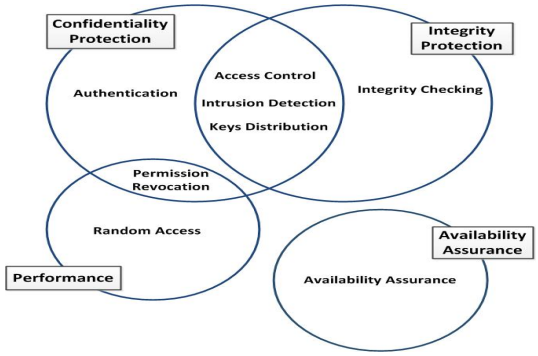

Fig. 2.: The relationship between the technologies and the design criteria.

VI. Conclusions

In this paper, we present a systematic survey on the research of secure storage. We describe the research background of securing data in the storage and give out the design criteria of secure storage systems. To make the research status be more easily understood, we extract the key technologies that are adopted in the existing secure storage systems, classify the technologies into different categories, and discuss their merits and weaknesses. Based on cloud storage, the new storage form which has drawn much attention recently, we also present the challenges of security in the cloud storage environment. Finally, we conclude and give the comparison of several typical secure storage systems.

VII. Acknowledgments

This work was supported by the National Science Foundation for Distinguished Young Scholars of China under Grant No.60925006, the National High Technology Research & Development Program of China under Grant No. 2013AA010101, and research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

[1] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou. Secure ranked keyword search over encrypted cloud data. in Proc. of ICDCS'10, 2010.

[2] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. Fuzzy keyword search over encrypted data in cloud computing. in Proc. of IEEE INFOCOM'10,2010.

[3] M. Li, S. Yu, N. Cao and W. Lou. Authorized Private Keyword Search over Encrypted Personal Health Records in Cloud Computing. In Proc. of ICDCS'11,2011.

[4] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. In Proc. of IEEE INFOCOM'11, 2011.

[5] E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multidimensional range query over encrypted data. In Proc. of IEEE S&P '07, 2007.

[6] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, *SiRiUS: Securing Remote Untrusted Storage*, In Proc. of NDSS'03, 2003.

[7] R. C. Merkle, *A digital signature based on a conventional encryption function*, In CRYPTO, volume 293, pages 369-378, 1987.

[8] E. Geron, A. Wool, *CRUST: Cryptographic remote untrusted storage without public keys*, In Proc. of IEEE SISW'07, 2007.

[9] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, *Plutus-scalable secure file sharing on untrusted storage.*,In Proc. of FAST'03, 2003.

[10] S. Kamara, and K. Lauter,*Cryptographic Cloud Storage*, In Proc. of FC'10,2010.

[11] M. Chase. *Multi-authority attribute based encryption*. In Theory of Cryptography Conference (TCC '07), volume 4392 of Lecture Notes in Computer Science, pages 515-534. Springer, 2007

[12] J. Bethencourt, A. Sahai, and B. Waters,*Ciphertext-Policy Attribute-Based Encryption*. In Proc. of IEEE S&P'07, 2007.

[13] V. Goyal, O. Pandey, A. Sahai, and B. Waters. *Attribute-based encryption for fine-grained access control of encrypted data*. In Proc. of ACM CCS'06, 2006.

[14] W. Xue, J. Shu, Y. Liu, and M. Xue. *Corslet: A shared storage system keeping your data private*. In SCIENCE CHINA Information Sciences, 2011, pp.1119-1128.

[15] A. Adya, W. Bolosky, M. Castro, G. Celmak, R. Chaiken, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. *FARSITE: Federated, available, and reliable storage for an incompletely trusted environment*. In Proc. of OSDI'02, 2002.

[16] P. Stanton. *Securing Data in Storage: A Review of Current Research*. CoRR, cs.OS/0409034, 2004.

[17] M. Pirretti, P. Traynor, P. Mcdaniel, and B.Waters. *Secure attribute-based systems*. In Proc. of ACM CCS'06, 2006.

[18] M. Chase. *Multi-Authority attribute based encryption*. In Proc. of TCC'07, 2007.

[19] J. Li, K. Ren, B. Zhu, Z. Wan. *Privacy-Aware attribute-based encryption with user accountability*. In Proc. of ISC'09, 2009.

[20] G. Ganger and D. Nagle. *Better Security via Smarter Devices*. In proc. of IEEE HOTOS'01, 2001.

[21] G. Ganger, J. Strunk, and A. Klosterman. *Self-storage: Brick-based storage with automated administration*. Carnegie Mellon University Technical Report, 2003.

[22] J. Molina, and W. Arbaugh. *Using Independent Auditors as Intrusion Detection Systems*. In Proc. of ICICS'02, 2002.

[23] Z. Li, Z. Chen, S. Srinivasan, and Y. Zhou *C-Miner: Mining Block Correlations in Storage Systems*. In Proc. of FAST'04, 2004.

[24] Y. Simmhan, B. Plale and D. Gannon. *A Survey of Data Provenance Techniques*. Technical Report IUB-CS-TR618, 2005.

[25] K. M-Reddy, P. Macko and M. Seltzer *Provenance for the cloud*. In Proc of FAST'10, 2010.

[26] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. *Provable data possession at untrusted stores*. In Proc. of ACM CCS'07, 2007.

[27] C. Wang, Q. Wang, K. Ren, and W. Lou. *Privacy-preserving public auditing for data storage security in cloud computing*. In Proc. of INFOCOM'10, 2010.

[28] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li. *Enabling public auditability and data dynamics for storage security in cloud computing*. Journal IEEE Transaction on Parallel and Distributed System, Volume 22 Issue 5, 2011.

[29] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia. *Dynamic provable data possession.*. In Proc. of ACM CCS'09, 2009.

[30] R. Curtmola, O. Khan, R. Burns, and G. Ateniese. *MR-PDP: Multiple-Replica provable data possession*. In Proc. of ICDCS'08, 2008.

[31] A. Juels, B. JR. *Pors: proofs of retrievability for large files*. In Proc. of ACM CCS'07, 2007.

[32] R. Perlman. *File system design with assured delete*. In Proc. of the Third IEEE International Security in Storage Workshop, 2007

[33] Y. Tang, P. Patrick and C. Lee, J. Lui, and R. Perlman. *FADE: Secure overlay cloud storage with file assured deletion*. In Proc. of the 6th International Conference on Security and Privacy in Communication Networks, 2010

[34] R. Geambasu, T. Kohno, A. Levy, and H. Levy. *Vanish: Increasing data privacy with self-destructing data*. In Proc. of USENIX Security'09, 2009.

[35] S. Wolchok, O, Hofmann, N. Heninger, E. Felten, J. Halderman, C. Rossbach, B. Waters, and E. Witchel. *Defeating vanish with low-cost sybil attacks against large DHTs*. In Proc. of NDSS'10, 2010.

[36] W. Storer, K. Greenan, D. Long, and E. Miller. *Secure data deduplication*. In Proc. of the 4th ACM International Workshop on Storage Security and Survivability, 2008.

[37] M. Mulazzani, S. Schrittwieser, M. Leithner, M. Huber, and E. Weippl. *Dark clouds on the horizon: using cloud storage as attack vector and online slack space*. In Proc. of SEC'11, 2011.

[38] V. Kher and Y. Kim. *Securing distributed storage: challenges, techniques, and systems*. In Proc. of StorageSS'05, 2005.

[39] J. Gamtz, C. Chute, A. Manfrediz, S. Minton, D. Reinsel, W. Schlichting, and A. Toncheva. *The diverse and exploding digital universe: an updated forecast of worldwide information growth through 2011*. IDC white paper, 2008.

[40] A. Singh, and Ling Liu. *SHAROES: A Data Sharing Platform for Outsourced Enterprise Storage Environments*. In Proc. of ICDE'08, 2008.

[41] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket. *Venus: Verification for Untrusted Cloud Storage.* In Proc. of CCSW'10, 2010.

[42] Z. Wilcox-O'Hearn, and B. Warner. *Tahoe-The Least-Authrity Filesystem.* In Proc. of StorageSS'08, 2008.

[43] J. Li, M. Krohn, D. Mazières, and D. Shasha. *Secure Untrusted Data Repository(SUNDR).* In Proc. of OSDI'04, 2004.

[44] M. Li, S. Yu, N. Cao and W. Lou. *Authorized Private Keyword Search over Encrypted Personal Health Records in Cloud Computing*. In Proc. of ICDCS'11,2011.

[45] P. Golle, J. Staddon, and B. Waters. *Secure conjunctive keyword search over encrypted data*. in Proc. of ACNS'04, 2004.