Toward Publicly Auditable Secure Cloud Data Storage Services

Cong Wang and Kui Ren, Illinois Institute of Technology
Wenjing Lou, Worcester Polytechnic Institute
Jin Li, Illinois Institute of Technology

Abstract

Cloud computing is the long dreamed vision of computing as a utility, where data owners can remotely store their data in the cloud to enjoy on-demand high-quality applications and services from a shared pool of configurable computing resources. While data outsourcing relieves the owners of the burden of local data storage and maintenance, it also eliminates their physical control of storage dependability and security, which traditionally has been expected by both enterprises and individuals with high service-level requirements. In order to facilitate rapid deployment of cloud data storage service and regain security assurances with outsourced data dependability, efficient methods that enable on-demand data correctness verification on behalf of cloud data owners have to be designed. In this article we propose that publicly auditable cloud data storage is able to help this nascent cloud economy become fully established. With public auditability, a trusted entity with expertise and capabilities data owners do not possess can be delegated as an external audit party to assess the risk of outsourced data when needed. Such an auditing service not only helps save data owners' computation resources but also provides a transparent yet cost-effective method for data owners to gain trust in the cloud. We describe approaches and system requirements that should be brought into consideration, and outline challenges that need to be resolved for such a publicly auditable secure cloud storage service to become a reality.

Cloud computing has been envisioned as the next-generation architecture of the IT enterprise due to its long list of unprecedented advantages in IT: on-demand self-service, ubiquitous network access, location-independent resource pooling, rapid resource elasticity, usage-based pricing, and transference of risk [1]. One fundamental aspect of this new computing model is that data is being centralized or outsourced into the cloud. From the data owners' perspective, including both individuals and IT enterprises, storing data remotely in a cloud in a flexible on-demand manner brings appealing benefits: relief of the burden of storage management, universal data access with independent geographical locations, and avoidance of capital expenditure on hardware, software, personnel maintenance, and so on [2].

While cloud computing makes these advantages more appealing than ever, it also brings new and challenging security threats to the outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing actually relinquishes the owner's ultimate control over the fate of their data. As a result, the correctness of the data in the cloud is put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they still face a broad range of both internal and external threats to data integrity. Outages and security breaches of noteworthy cloud services appear from time to time. Amazon S3's recent downtime [3], Gmail's mass email deletion incident [4], and Apple MobileMe's post-launch downtime [5] are all such examples. Second, for benefits of their own, there are various motivations for CSPs to behave unfaithfully toward cloud customers regarding the status of their outsourced data. Examples include CSPs, for monetary reasons, reclaiming storage by discarding data that has not been or is rarely accessed [6], or even hiding data loss incidents to maintain a reputation [7]. In short, although outsourcing data into the cloud is economically attractive for the cost and complexity of long-term large-scale data storage, it does not offer any guarantee on data integrity and availability. This problem, if not properly addressed, may impede successful deployment of the cloud architecture.

As data owners no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted [6, 7]. In particular, simply downloading the data for its integrity verification is not a practical solution due to the high cost of input/output (I/O) and transmission across the network. Besides, it is often insufficient to detect data corruption only when accessing the data, as it does not give correctness assurance for unaccessed data and might be too late to recover the data loss or damage. Considering the large size of the outsourced data and the owner's constrained resource capability, the tasks of auditing the data correctness in a cloud environment can be formidable and expensive for data owners [6–8]. Moreover, from the system usability point of view, data owners should be

 0890-8044/10/$25.00 © 2010 IEEE

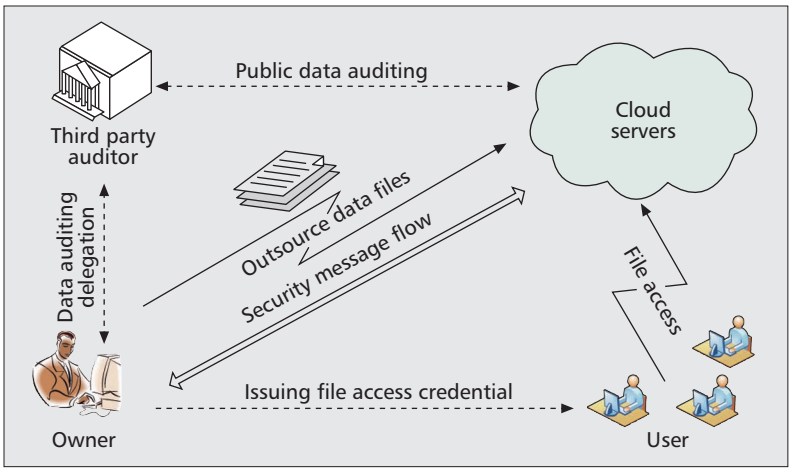

Figure 1. *The architecture of cloud data storage service.*

able to just use cloud storage as if it is local, without worrying about the need to verify its integrity. Hence, to fully ensure data security and save data owners' computation resources, we propose to enable publicly auditable cloud storage services, where data owners can resort to an external third party auditor (TPA) to verify the outsourced data when needed. Third party auditing provides a transparent yet cost-effective method for establishing trust between data owner and cloud server. In fact, based on the audit result from a TPA, the released audit report would not only help owners to evaluate the risk of their subscribed cloud data services, but also be beneficial for the cloud service provider to improve their cloud based service platform [8]. In a word, enabling public risk auditing protocols will play an important role for this nascent cloud economy to become fully established, where data owners will need ways to assess risk and gain trust in the cloud.

Recently, great interest has been shown in ensuring remotely stored data integrity under different system and security models [6–15]. Some of the work [7, 8, 10, 13, 15] has already been promoting the development of public auditability for existing cloud data storage services. However, it is not feasible yet. On one hand, data owners are currently not sophisticated enough to demand risk assessment; on the other hand, current commercial cloud vendors do not provide such a third party auditing interface to support a public auditing service. This article is intended as a call for action, aiming to motivate further research on dependable cloud storage services and enable public auditing services to become a reality. We start by suggesting a set of systematically and cryptographically desirable properties that should apply to practical deployment for securing the cloud storage on behalf of data owners. We sketch a set of building blocks, including recently developed cryptographic primitives (e.g., homomorphic authenticator), to ensure these strong security properties, which could form the basis of a publicly auditable secure cloud data storage system.

Cloud Storage Architecture and Security Threats

Problem Statement

We begin with a high-level architecture description of cloud data storage services illustrated in Fig. 1. At its core, the architecture consists of four different entities: *data owner*, *user*, *cloud server* (CS), and *TPA*. Here the TPA is the trusted entity that has expertise and capabilities to assess cloud storage security on behalf of a data owner upon request. Under the cloud paradigm, the data owner may represent either the individual or the enterprise customer, who relies on the cloud server for remote data storage and maintenance, and thus is relieved of the burden of building and maintaining local storage infrastructure. In most cases cloud data storage services also provide benefits like availability (being able to access data from anywhere), relative low cost (paying as a function of need), and on-demand sharing among a group of trusted users, such as partners in a collaboration team or employees in the enterprise organization. For simplicity, we assume a single writer/many readers scenario here. Only the data owner can dynamically interact with the CS to update her stored data, while users just have the privilege of file reading.

Within the scope of this article, we focus on how to ensure publicly auditable secure cloud data storage services. As the data owner no longer possesses physical control of the data, it is of critical importance to allow the data owner to verify that his data is being correctly stored and maintained in the cloud. Considering the possibly large cost in terms of resources and expertise, the data owner may resort to a TPA for the data auditing task to ensure the storage security of her data, while hoping to keep the data private from the TPA. We assume the TPA, who is in the business of auditing, is reliable and independent, and thus has no incentive to collude with either the CS or the owners during the auditing process. The TPA should be able to efficiently audit the cloud data storage without local copy of data and without any additional online burden for data owners. Besides, any possible leakage of an owner's outsourced data toward a TPA through the auditing protocol should be prohibited.

We consider both malicious outsiders and a semi-trusted CS as potential adversaries interrupting cloud data storage services. Malicious outsiders can be economically motivated, and have the capability to attack cloud storage servers and subsequently pollute or delete owners' data while remaining undetected. The CS is semi-trusted in the sense that most of the time it behaves properly and does not deviate from the prescribed protocol execution. However, for its own benefit the CS might neglect to keep or deliberately delete rarely accessed data files that belong to ordinary cloud owners. Moreover, the CS may decide to hide the data corruptions caused by server hacks or Byzantine failures to maintain its reputation. Note that in our architecture, we assume that basic security mechanisms such as a preloaded public/private key pair with each entity are already in place to provide basic communication security, which can be achieved in practice with little overhead.

Desirable Properties for Public Auditing

Our goal is to enable public auditing for cloud data storage to become a reality. Thus, the whole service architecture design should not only be cryptographically strong, but, more important, be practical from a systematic point of view. We briefly elaborate a set of suggested desirable properties below that satisfy such a design principle. The in-depth analysis is discussed in the next section. Note that these requirements are ideal goals. They are not necessarily complete yet or even fully achievable in the current stage.

Minimize Auditing Overhead — First and foremost, the overhead imposed on the cloud server by the auditing process must not outweigh its benefits. Such overhead may include both the I/O cost for data access and the bandwidth cost for data transfer. Any extra online burden on a data owner

	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]	[15]
Constant bandwidth cost	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
No online burden on data owner		✓			✓	✓		✓	✓	✓
Protecting data privacy	✓[2]		✓[2]	✓						✓
Data dynamics support				✓[1]			✓[1]	✓	✓	✓
Batch auditing					✓[3]			✓[3]		✓

[1] The schemes only support partially data updates; that is, data insertion is not supported.

[2] The schemes build on the encrypted data only.

[3] No explicit implementation of batch auditing is given for these schemes.

Table 1. *Various schemes and their support of different requirements.*

should also be as low as possible. Ideally, after auditing delegation, the data owner should just enjoy the cloud storage service while being worry-free about storage auditing correctness.

Protect Data Privacy — Data privacy protection has always been an important aspect of a service level agreement for cloud storage services. Thus, the implementation of a public auditing protocol should not violate the owner's data privacy. In other words a TPA should be able to efficiently audit the cloud data storage without demanding a local copy of data or even learning the data content.

Support Data Dynamics — As a cloud storage service is not just a data warehouse, owners are subject to dynamically updating their data via various application purposes. The design of auditing protocol should incorporate this important feature of data dynamics in Cloud Computing.

Support Batch Auditing — The prevalence of large-scale cloud storage service further demands auditing efficiency. When receiving multiple auditing tasks from different owners' delegations, a TPA should still be able to handle them in a fast yet cost-effective fashion. This property could essentially enable the scalability of a public auditing service even under a storage cloud with a large number of data owners.

Ensuring Cloud Data Security

In this section we start from scratch and a set of building blocks that could form the basis of public auditing services for dependable cloud data storage. For some of the building blocks, we can rely on existing work or newly developed cryptographic primitives; for others, we only sketch the problem and leave it for future research. Table 1 gives the state of the art and their support of different requirements.

Traditional Methods Revisited

A straightforward approach to protect the data integrity would be using traditional cryptographic methods, such as the well-known message authentication codes (MACs). Initially, data owners can locally maintain a small amount of MACs for the data files to be outsourced. Whenever the data owner needs to retrieve the file, she can verify the integrity by recalculating the MAC of the received data file and comparing it to the locally precomputed value. If the data file is large, a hash tree [16] can be employed, where the leaves are hashes of data blocks and internal nodes are hashes of their children of the tree. The data owner only needs to store the root hash of the tree to authenticate his received data.

While this method allows data owners to verify the correctness of the received data from the cloud, it does not give any assurance about the correctness of other outsourced data. In other words, it does not give any guarantee that the data in the cloud are all actually intact, unless the data are all downloaded by the owner. Because the amount of cloud data can be huge, it would be quite impractical for a data owner to retrieve all of her data just in order to verify the data is still correct. If the data auditing task is delegated to a TPA, this method inevitably violates our suggested requirements, with large auditing cost for a cloud server (for accessing and transferring all of the data) and data privacy exposure to the TPA (for retrieving a local copy of data). Thus, new approaches are required.

To avoid retrieving data from the cloud server, a simple improvement to this straightforward solution can be performed as follows: Before data outsourcing, the owner chooses a set of random MAC keys, precomputes the MACs for the whole data file, and publishes these verification metadata to the TPA. The TPA can each time reveal a secret MAC key to the cloud server and ask for a fresh keyed MAC for comparison. In this way the bandwidth cost for each audit is only at the bit-length level (keys and MACs). However, a particular drawback is that the number of times a data file can be audited is limited by the number of secret keys that must be fixed a priori, which might introduce an additional online burden to the data owner: Once all possible secret keys are exhausted, the data owner then has to retrieve data from the server in order to recompute and republish new MACs to the TPA. Another drawback of this improved approach is its inability to deal with data dynamics, as any data change would make those precomputed MACs completely unusable.

Utilizing Homomorphic Authenticators

To significantly reduce the arbitrarily large communication overhead for public auditability without introducing any online burden on the data owner, we resort to the homomorphic authenticator technique [7, 10]. Homomorphic authenticators are unforgeable metadata generated from individual data blocks, which can be securely aggregated in such a way to assure a verifier that a linear combination of data blocks is correctly computed by verifying only the aggregated authenticator.

Using this technique requires additional information encoded along with the data before outsourcing. Specifically, a data file is divided into n blocks m_i ($i = 1, \ldots, n$), and each block m_i has a corresponding homomorphic authenticator σ_i computed as its metadata to ensure the integrity. Every time it must be

verified that the cloud server is honestly storing the data, the data owner or TPA can submit challenges $chal = \{(i, \nu_i)\}$ for sampling a set of randomly selected blocks, where $\{\nu_i\}$ can be arbitrary weights. Due to the nice property of the homomorphic authenticator, server only needs to response a linear combination of the sampled data blocks $\mu = \sum_i \nu_i \cdot m_i$, as well as an aggregated authenticator $\sigma = \prod_i \sigma_i^{\nu_i}$, both computed from $\{m_i, \sigma_i, \nu_i\}_{i \in chal}$. Once the response of μ and σ is verified by TPA, then high probabilistic guarantee on large fraction of cloud data correctness can be obtained.[1] Because off-the-shelf error-correcting code technique can be adopted before data outsourcing [6, 10], large fraction of correct cloud data would be sufficient to recover the whole data.

Note that for typical choices of block size $|m_i|$ and file block number n, where $|m_i| \gg \log(n)$, the response μ and σ are (essentially) about the same size as individual block m_i and σ_i. This means almost constant communication overhead, independent of file size, for each auditing can be achieved. Moreover, since the TPA could *regenerate* the fresh random sampling challenges, unbounded auditing is achieved too, which means no additional on-line burden would be incurred towards data owner. However, despite the desirable properties, this approach only works well for encrypted data. When directly applied to unencrypted data, it still leaks bits information towards TPA, as discussed next.

Protecting Data Privacy

The reason that linear combination of sampled blocks may potentially reveal owner data information is due to the following fact about basic linear algebra theory: if enough linear combinations of the same blocks are collected, the TPA can simply derive the sampled data content by solving a system of linear equations.

This drawback greatly affects the security of using homomorphic-authenticator-based techniques in a publicly auditable cloud data storage system. From the perspective of protecting data privacy, the owners, who own the data and rely on the TPA just for the storage security of their data, do not want this auditing process introducing new vulnerabilities of unauthorized information leakage into their data security [8, 15]. Moreover, there are legal regulations, such as the U.S. Health Insurance Portability and Accountability Act (HIPAA) [17], further demanding the outsourced data not to be leaked to external parties. Exploiting data encryption before outsourcing [6, 8] is one way to mitigate this privacy concern, but it is only complementary to the privacy-preserving public auditing scheme to be deployed in cloud. Without a properly designed auditing protocol, encryption itself can not prevent data from *flowing away* toward external parties during the auditing process. Thus, it does not completely solve the problem of protecting data privacy but just reduces it to the one of managing the encryption keys. Unauthorized data leakage still remains a problem due to the potential exposure of encryption keys.

To address this concern, a proper approach is to combine the homomorphic authenticator with random masking. This way, the linear combination of sampled blocks in the server's response is masked with randomness generated by the server. With random masking, the TPA no longer has all the necessary information to build up a correct group of linear equations and therefore cannot derive the owner's data content, no matter how many linear combinations of the same set of file blocks can be collected. Meanwhile, due to the algebraic property of the homomorphic authenticator, the correctness validation of the block-authenticator pairs (μ and σ) can still be carried out in a new way, even in the presence of randomness. Some initial results of this branch of research can be found in [15].

This improved technique ensures the privacy of owner data content during the auditing process, regardless of whether or not the data is encrypted, which definitely provides more flexibility for different application scenarios of cloud data storage. Besides, with the homomorphic authenticator, the desirable property of constant communication overhead for the server's response during the audit is still preserved.

Supporting Data Dynamics

Cloud computing is not just a third party data warehouse. For various application purposes, the data stored in the cloud may not only be accessed but also updated frequently by data owners [9, 12–14]. Thus, supporting data dynamics, which can be formulated as general block-level operations, including block modification, block insertion, and block deletion, is also of critical importance for auditing mechanisms under the cloud data storage paradigm.

Using homomorphic authenticators helps achieve a constant communication overhead for public auditability. However, the direct extension of the approach to support data dynamics may have security and efficiency problems [13]. Take block insertion, for example. In the original homomorphic authenticator schemes, to prevent a cloud server using the same authenticator to obtain the correctness proof for a different block, the block index information i has to be embedded in the authenticator calculation (usually in the form of $H(i)$, where $H(\cdot)$ is a collision resistant hash function). As a result, any insertion operation (e.g., inserting block m^* after m_i) will inevitably change the indices of all the following blocks after i, causing significantly expensive recomputation of all the corresponding authenticators. In order to avoid the above dilemma for data dynamics, we have to find a way to eliminate the index information in the homomorphic authenticator calculation while not affecting the security.

To satisfy this special requirement, the first intuition would be using authenticated data structures, such as the well studied Merkle hash tree (MHT) [16], which is intended to efficiently and securely prove that a set of elements is undamaged and unaltered. By treating the leaf nodes of the MHT as the file blocks m_i, we immediately achieve easy verification of m_i with respect to a publicly known root value $\mathcal{R}$ and the auxiliary authentication information (AAI) of the very leaf, which includes the siblings of the nodes on the path connecting the leaf to the root. However, directly using these structures shares the same disadvantages of the straightforward scheme:

- It requires a linear amount of communication cost with respect to the number of sampled data size, which can be arbitrarily large.
- It introduces vulnerabilities of owner data privacy when applied to unencrypted data directly.

To further support secure and efficient data operations, one approach would be integrating the MHT with the homomorphic-authenticator-based technique [13], taking advantage of both. (Note that it is possible to use other authenticated data structures instead of the MHT, like an authenticated skip list as in [14].) The hybrid approach is achieved by removing the index information $H(i)$ in the authenticator construction and treating the MHT leave nodes with the newly computed authenticator instead of m_i. In doing so, the integrity of the authenticator themselves is protected by the MHT, while the authenticators further protect the integrity of the blocks. This gives two immediate advantages:

[1] *Usually sampling several hundred blocks would be enough when 1 percent of a data file is corrupted [7].*

- The individual data operation on any file block, especially block insertion and deletion, will no longer affect other unchanged blocks.
- Blockless auditing is still preserved: for each auditing process, the cloud server can still send back a small-sized linear combination of blocks $\mu = \sum_i \nu_i m_i$ and an aggregated authenticator σ, imposing much smaller communication overhead than sending individual blocks.

Also, we should notice that this integrated approach can be further combined with the random masking technique, realizing data privacy protection.

Handling Multiple Concurrent Tasks

With the establishment of privacy-preserving public auditing in cloud computing, a TPA may concurrently handle auditing delegations on different owners' requests. The individual auditing of these tasks in a sequential way can be tedious and very inefficient for a TPA. Given K auditing delegations on K distinct data files from K different owners, it is more advantageous for a TPA to batch these multiple tasks together and perform the auditing one time, saving computation overhead as well as auditing time cost.

Keeping this natural demand in mind, we note that two previous works [10, 13] can be directly extended to provide batch auditing functionality by exploring the technique of bilinear aggregate signature [18]. Such a technique supports the aggregation of multiple signatures by distinct signers on distinct messages into a single signature and thus allows efficient verification for the authenticity of all messages. Basically, with batch auditing the K verification equations (for K auditing tasks) corresponding to K responses $\{\mu, \sigma\}$ from a cloud server can now be aggregated into a single one such that a considerable amount of auditing time is expected to be saved. A very recent work [15] gives the first study of batch auditing and presents mathematical details as well as security reasonings.

Note that the aggregated verification equation in batch auditing only holds when all the responses are valid, and fails with high probability when there is even one single invalid response in the batch auditing. To further sort out these invalid responses, a recursive divide-and-conquer approach (binary search) can be utilized. Specifically, if the batch auditing fails, we can divide the collection of responses into two halves, and recurse the batch auditing in halves. Preliminary results in [15] shows that compared to individual auditing, batch auditing indeed helps reduce the TPA's computation cost, as more than 11 and 14 percent of per-task auditing time is saved when the sampling block set is set to be 460 and 300, respectively. Moreover, even if up to 18 percent of 256 different responses are invalid, batch auditing still performs faster than individual verification.

Further Challenges

In the above sections we have described some suggested requirements for public auditing services and the state of the art that fulfills them. However, this is still not enough for a publicly auditable secure cloud data storage system, and further challenging issues remain to be supported and resolved.

Accountability — From the viewpoint of the threat model, all above discussions only regard the cloud server as untrusted, and the TPA's auditing result only indicates the honestness of cloud server. But what if the data owner and/or the TPA are untrustworthy as well? In this case the auditing result should not only identify the data correctness but also be able to determine which entity (including owner, TPA, as well as cloud server) is responsible for the problem that may occur. It is not yet clear how to achieve entity accountability when all entities are possibly malicious.

Multi-Writer Model — As mentioned, cloud data storage not only provides dynamic and scalable storage services, but also allows easy on-demand file sharing. A difficult problem is support for services with legacy users, who may not only access but also *modify* the owner's data in the cloud. Under this multi-writer model, achieving the same data dynamics support for public auditing services while maintaining file consistency is another future challenge.

Performance — Performance is always an important concern for practical system deployment. Although there is evidence that the overhead for auditing based on homomorphic authenticators will be manageable [13–15], we have yet to demonstrate that the cost of authenticator precomputation and transfer of a realistic personal device is acceptable.

Concluding Remarks

Cloud computing has been envisioned as the next-generation architecture of enterprise IT. In contrast to traditional enterprise IT solutions, where the IT services are under proper physical, logical, and personnel controls, cloud computing moves the application software and databases to servers in large data centers on the Internet, where the management of the data and services are not fully trustworthy. This unique attribute raises many new security challenges in areas such as software and data security, recovery, and privacy, as well as legal issues in areas such as regulatory compliance and auditing, all of which have not been well understood. In this article we focus on cloud data storage security. We first present a network architecture for effectively describing, developing, and evaluating secure data storage problems. We then suggest a set of systematically and cryptographically desirable properties for public auditing services of dependable cloud data storage security to become a reality. Through in-depth analysis, some existing data storage security building blocks are examined. The pros and cons of their practical implications in the context of cloud computing are summarized. Further challenging issues for public auditing services that need to be focused on are discussed too. We believe security in cloud computing, an area full of challenges and of paramount importance, is still in its infancy now but will attract enormous amounts of research effort for many years to come.

References

[1] P. Mell and T. Grance, "Draft NIST Working Definition of Cloud Computing," 2009; http://csrc.nist.gov/groups/SNS/cloud-computing/index.html
[2] M. Armbrust *et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," Univ. California, Berkeley, Tech. Rep. UCBEECS-2009-28, Feb. 2009.
[3] Amazon.com, "Amazon s3 Availability Event: July 20, 2008," July 2008; http://status.aws.amazon.com/s3-20080720.html
[4] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," Dec. 2006; http://www.techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions/
[5] M. Krigsman, "Apple's MobileMe Experiences Post-Launch Pain," July 2008; http://blogs.zdnet.com/projectfailures/?p=908
[6] A. Juels, J. Burton, and S. Kaliski, "PORs: Proofs of Retrievability for Large Files," *Proc. ACM CCS '07*, Oct. 2007, pp. 584–97.
[7] G.Ateniese *et al.*, "Provable Data Possession at Untrusted Stores," *Proc. ACM CCS '07*, Oct. 2007, pp. 598–609.
[8] M. A. Shah *et al.*, "Auditing to keep Online Storage Services Honest," *Proc. USENIX HotOS '07*, May 2007.
[9] G. Ateniese *et al.*, "Scalable and Efficient Provable Data Possession," *Proc. SecureComm '08*, Sept. 2008.
[10] H. Shacham and B. Waters, "Compact Proofs of Retrievability," *Proc. Asia-Crypt '08*, LNCS, vol. 5350, Dec. 2008, pp. 90–107.
[11] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A High-Availability and Integrity Layer for Cloud Storage," *Proc. ACM CCS '09*, Nov. 2009, pp. 187–98.

[12]C.Wang *et al.*,"Ensuring Data Storage Security in Cloud Computing," *Proc. IWQoS '09*, July 2009, pp. 1–9.
[13] Q. Wang *et al.*, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing," *Proc. ESORICS '09*, Sept. 2009, pp. 355–70.
[14] C. Erway *et al.*, "Dynamic Provable Data Possession," *Proc. ACM CCS '09*, Nov. 2009, pp. 213–22.
[15] C. Wang *et al.*, "Privacy-Preserving Public Auditing for Storage Security in Cloud Computing," *Proc. IEEE INFOCOM '10*, Mar. 2010.
[16] R. C. Merkle, "Protocols for Public Key Cryptosystems," *Proc. IEEE Symp. Security Privacy*, 1980.
[17] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPAA)," 1996; http://aspe.hhs.gov/admnsimp/pl104191.htm
[18] D. Boneh *et al.*, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," *Proc. EuroCrypt '03*, LNCS, vol. 2656, May 2003, pp. 416–32.

Biographies

CONG WANG (cong@ece.iit.edu) received his B.E and M.E. degrees from Wuhan University, China, in 2004 and 2007, respectively. He is currently a Ph.D. student in the Electrical and Computer Engineering Department at Illinois Institute of Technology. His research interests are in the areas of applied cryptography and network security, with current focus on secure data services in cloud computing.

KUI REN (kren@ece.iit.edu) is an assistant professor in the Electrical and Computer Engineering Department at Illinois Institute of Technology. He obtained his Ph.D. degree in electrical and computer engineering from Worcester Polytechnic Institute in 2007. His research interests include network security and privacy and applied cryptography, with the current focus on security and privacy in cloud computing, lower-layer attack and defense mechanisms for wireless networks, e-healthcare, and sensor network security. His research is sponsored by the U.S. National Science Foundation. He is a member of ACM.

WENJING LOU (wjlou@ece.iit.edu) earned a Ph.D. in electrical and computer engineering at the University of Florida. She joined the Electrical and Computer Engineering Department at Worcester Polytechnic Institute as an assistant professor in 2003, where she is now an associate professor. Her current research interests are in the areas of ad hoc, sensor, and mesh networks, with emphases on network security and routing issues. She was a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) award in 2008.

JIN LI (jin.li@ece.iit.edu) received his B.S. degree in mathematics from Southwest University in 2002, and his Ph.D. degree from Sun Yat-sen University in 2007. From 2009 to 2010 he was an associate senior researcher in the Electrical and Computer Engineering Department at Illinois Institute of Technology. In 2010 he joined the faculty of the Department of Computer Science, Guangzhou University. His current research interests include cryptography and security in cloud computing.