SCIENCE CHINA
Information Sciences

• LETTER •

May 2019, Vol. 62 059101:1–059101:3
https://doi.org/10.1007/s11432-018-9500-0

An efficient $i\mathcal{O}$-based data integrity verification scheme for cloud storage

Lixue SUN[1*], Chunxiang XU[1*], Yuan ZHANG[1,2] & Kefei CHEN[3]

[1]*Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 611731*, China;*
[2]*Department of Electrical and Computer Engineering, University of Waterloo, Waterloo* N2L 3G1*, Canada;*
[3]*School of Science, Hangzhou Normal University, Hangzhou* 310036*, China*

Received 12 April 2018/Accepted 20 June 2018/Published online 12 October 2018

Citation Sun L X, Xu C X, Zhang Y, et al. An efficient $i\mathcal{O}$-based data integrity verification scheme for cloud storage. Sci China Inf Sci, 2019, 62(5): 059101, https://doi.org/10.1007/s11432-018-9500-0

Dear editor,
With the advent of big data, an increasing number of people are storing their data on remote cloud servers. Cloud storage services enable users to flexibly maintain and access their data via mobile or wireless networks. However, despite these benefits, cloud storage deprives users of physical control over their data, which poses new challenges in data security [1–4]. Data integrity protection is one of the most important of these security challenges.

In existing public data integrity verification schemes, users are free from the burden of data verification as an external third-party auditor (TPA) is employed to periodically verify data integrity. However, most existing public data integrity verification schemes incur excessive storage overhead and data processing costs on the user side. These schemes utilize homomorphic linear authentication, where each data block is accompanied by an authenticator of equal size. Moreover, in these schemes, the user must compute an authenticator for each data block; thus, too many public key cryptographic operations are required to process the data. To solve these problems, we employ a cryptographic primitive indistinguishability obfuscator ($i\mathcal{O}$) [5] to design a public data integrity verification scheme; specifically, users' secret information will be embedded into an obfuscated program and hidden from the cloud server. Numerous cryptographic applications have recently been constructed based on $i\mathcal{O}$ [6–8].

The proposed data integrity verification scheme comprises three elements: a cloud user, a TPA, and a cloud server. The cloud user uploads his data files to the cloud server for storage. Users periodically audit their files to ensure data integrity. However, users are generally provided with limited computation resources; the TPA have both professional expertise and abundant resources with which they perform audit operations. Thus, users will refer to the TPA to check the integrity of their data; however, users generally wish to keep their data secret from the TPA. Moreover, the cloud server is assumed to be self-interested in the proposed scheme; hence, it may be compromised by outside attackers (usually for profit). For example, the cloud server may maliciously delete data files and claim that the data are intact.

Here, we present an overview of our data integrity verification scheme. We assumed that the TPA could verify a signature using a hash value of the user's data to perform an audit. The server was assumed to be semi-trusted in our scheme, as it may forge a proof to deceive the TPA. We employed $i\mathcal{O}$ to resist such misbehavior from the server. In addition, we applied the punctured programs technique to guarantee the security of our

* Corresponding author (email: sunlixue2007@sina.com, chxxu@uestc.edu.cn)

scheme; specifically, we introduced a puncturable pseudorandom function (PRF) that uses the hash value of the user's stored data as an input. Thus, the TPA will verify the signature on the output of the PRF to check data integrity. Moreover, to resist replay attacks launched by the server, a random element will be selected as the challenge and sent to the server by the TPA.

In our scheme, a user first generates an obfuscated program that embeds his/her signing key and the key for the PRF and uploads it to the server. Next, the user calculates the PRF with a hash value of user's data and forwards the output of the PRF to the TPA. Following this, the user sends his/her data to the server, and the server executes the obfuscated program upon receiving a challenge message from the TPA. Only when the outsourced data is preserved and intact will the obfuscated program produce a signature on the data. Finally, the TPA checks the data integrity by verifying the validity of the signature. Note that the obfuscated program is dependent on the security parameters, user's signing key, and PRF key, and will not be changed with the change of subsequent inputs. Thus, to generate this obfuscated program requires a one-time cost.

For more details on $i\mathcal{O}$ and puncturable PRFs, please refer to Appendixes A and B.

The definitions of the proposed data integrity verification scheme are as follows.

Definition 1. The proposed data integrity verification scheme consists of the following three algorithms.

Setup(1^λ)**.** Taking the security parameter λ as input, this randomized algorithm outputs the secret key sk and the public parameter pp.

Store(sk, F)**.** This algorithm takes a data file F and secret key sk as input, generates an authentication information h on F and a tag t, and sends tuples (t, h) and (t, F) to the TPA and cloud server, respectively.

Audit$(\mathrm{pp}, \mathrm{chal}, F, h)$**.** The inputs of this algorithm are pp, a challenge message chal, a data file F, and authentication information h. The cloud server computes the response message Prf using F and chal. Following this, Prf is sent to the TPA by the cloud server. Finally, the integrity of F is checked by verifying Prf with pp and h on the TPA side. If verification is successful, the algorithm outputs 1, which indicates that the server has maintained the file F; otherwise, the algorithm outputs 0.

Correctness. For all public parameters pp and secret keys sk generated by $\mathrm{Setup}(1^\lambda)$, (t, h) generated by $\mathrm{Store}(\mathrm{sk}, F)$, files F, and challenge messages chal, the algorithm $\mathrm{Audit}(\mathrm{pp}, \mathrm{chal}, F, h)$ outputs 1 with respect to valid responses Prf.

Our construction. As previously mentioned, our basic scheme comprises three elements: a cloud server $\mathcal{C}$, a TPA, and a cloud user $\mathcal{U}$. To define a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^l$ and a puncturable PRF $f : \{0,1\}^l \to \{0,1\}^\lambda$, let $\mathcal{S}$ be a secure deterministic signature scheme.

Setup(1^λ)**.** Within the security parameter λ, the system parameters are generated by the cloud user $\mathcal{U}$ as follows:

(1) Select a puncturable PRF key K for f, and a random verifying/signing key pair (svk, ssk) for $\mathcal{S}$;

(2) Generate one circuit $\mathrm{Audit}_{K,\mathrm{ssk}}$ as described in Figure 1 and compute $\mathcal{P} = i\mathcal{O}(\mathrm{Audit}_{K,\mathrm{ssk}})$, where Sig is the signature algorithm of $\mathcal{S}$ and r is a random element chosen by the TPA.

The public parameter is $\mathrm{pp} = (\mathrm{svk}, \mathcal{P})$, and the secret key is $\mathrm{sk} = (K, \mathrm{ssk})$.

$\mathrm{Audit}_{K,\mathrm{ssk}}$:
Input: F, r.
Constants: K, ssk.
Compute $h = f(K, H(F))$ and output
$\sigma = \mathrm{Sig}_{\mathrm{ssk}}(h \| r)$.

Figure 1 Audit circuit $\mathrm{Audit}_{K,\mathrm{ssk}}$.

Store(sk, F)**.** $\mathcal{U}$ stores the data file F with his/her secret key as follows:

(1) Select a random element name as the identifier of the data file F and compute $h = f(K, H(F))$;

(2) Send the tuples (name, h) and (name, F) to the TPA and $\mathcal{C}$, respectively.

Audit$(\mathrm{name}, r, F, \mathrm{svk}, h)$**.** There are three steps included in this algorithm:

(1) To audit the data file F, the TPA sends a challenge message (name, r) to $\mathcal{C}$, where r is a random element for each audit chosen by the TPA;

(2) Upon receiving (name, r), $\mathcal{C}$ first looks up the stored files and obtains the file F. Next, $\mathcal{C}$ computes the response proof $\mathrm{Prf} = \mathcal{P}(F, r)$ and sends the TPA the proof;

(3) When the TPA receives Prf, it checks the integrity of F by using svk to verify Prf.

If verification is successful, the algorithm outputs 1.

Correctness. For a challenge message (name, r), the valid response proof $\sigma = \mathrm{Sig}_{\mathrm{ssk}}(h \| r)$ output by the obfuscated program $\mathcal{P}$ is a signature on $h \| r$. Thus, the accuracy of the basic scheme follows that of $\mathcal{S}$.

Remark. If the outsourced data file is split into multiple blocks, the storage overhead of existing schemes is approximately double the file size. In contrast, only the data file itself is needed to be maintained by the cloud server in our scheme.

Furthermore, our scheme protects the privacy of the data file (an appealing feature); as the hash value of the data is input to a keyed PRF, the TPA cannot extract the file contents without the key for the PRF.

In addition to access, the user may want to update their data after moving them to the server. Supporting dynamic updates in an auditing scheme includes inserting, deleting, and modifying stored files. Our proposed scheme can efficiently support block level updates before moving it to the server. To update one data block, the cloud user needs only to re-evaluate the puncturable PRF on the hash value of the data and send it to the TPA.

Security. The security of our scheme includes authenticity and retrievability, which can be defined as in [9]. Authenticity requires that the server can always be detected by the user if it deviates from honest behavior. Retrievability guarantees that if a malicious server passes verification with non-negligible probability, it can access all data; moreover, the data can be retrieved from the server in polynomial time.

Assuming $\mathcal{S}$ is a deterministic signature scheme that satisfies existential unforgeability, f is a secure puncturable PRF, $i\mathcal{O}$ is a secure indistinguishability obfuscator, and H is a collision resistant hash function, our scheme satisfies authenticity and retrievability.

Conclusion. We propose a public data integrity verification scheme by applying $i\mathcal{O}$, which relieves users of the huge costs of calculating authenticators and significantly reduces storage overhead. Without splitting the data file into blocks, the cloud user in our scheme needs only to evaluate a puncturable PRF on their entire data set. Compared with existing data integrity verification schemes, the computation overhead of our scheme is significantly reduced on the user side. In addition, verification overhead on the TPA side in the majority of existing schemes increases noticeably with the size of audited data file; our scheme is effective regardless of file size. Finally, the privacy of user data will be maintained, thereby protecting the data against potential breaches from the TPA.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2017YFB0802000) and National Natural Science Foundation of China (Grant No. 61370203).

Supporting information Appendixes A and B. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Li H W, Liu D X, Dai Y S, et al. Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage. IEEE Trans Emerg Top Comput, 2015, 3: 127–138

2 Zhang Y, Xu C X, Yu S, et al. SCLPV: secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors. IEEE Trans Comput Soc Syst, 2015, 2: 159–170

3 Li H W, Yang Y, Dai Y S, et al. Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Trans Cloud Comput, 2017. doi: 10.1109/TCC.2017.2769645

4 Sun L X, Xu C X, Zhang Y. A dynamic and non-interactive boolean searchable symmetric encryption in multi-client setting. J Inf Secur Appl, 2018, 40: 145–155

5 Garg S, Gentry C, Halevi S, et al. Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of the 54th Annual Symposium on Foundations of Computer Science, Berkeley, 2013. 40–49

6 Zhang Y, Xu C X, Liang X H, et al. Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. IEEE Trans Inf Forensic Secur, 2017, 12: 676–688

7 Sun L X, Xu C X, Zhang M W, et al. Secure searchable public key encryption against insider keyword guessing attacks from indistinguishability obfuscation. Sci China Inf Sci, 2018, 61: 038106

8 Guan C W, Ren K, Zhang F G, et al. Symmetric-key based proofs of retrievability supporting public verification. In: Proceedings of European Symposium on Research in Computer Security, Vienna, 2015. 203–223

9 Shi E, Stefanov E, Papamanthou C. Practical dynamic proofs of retrievability. In: Proceedings of ACM SIGSAC Conference on Computer and Communications Security, Berlin, 2013. 325–336