Security analysis of DeyPoS scheme

Taek-Young Youn[1], Nam-Su Jho[1], Si-Wan Noh[2], Kyung-Hyune Rhee[3], and Sang Uk Shin[3]*
[1]Electronics and Telecommunications Research Institute (ETRI), Republic of Korea
{taekyoung, nsjho}@etri.re.kr
[2]Interdisciplinary Program of Information Security,
Graduate School, Pukyong National University, Republic of Korea
nosiwan@pukyong.ac.kr
[3]Dept. of IT Convergence and Application Engineering,
Pukyong National University, Republic of Korea
{khrhee, shinsu}@pknu.ac.kr

Abstract

Recently, He *et al*. [3], proposed a concrete scheme of deduplicatable dynamic PoS called DeyPoS, to achieve dynamic PoS (Proof of Storage) and secure cross-user deduplication, simultaneously. However, after a close look at their proposed scheme, we identity some security weaknesses. In this paper, we show the security vulnerability of DeyPoS scheme, compromise of data privacy and unforgeability.

Keywords: Deduplication, Cloud computing, Proof of Storage, Data Privacy, Unforgeability

1 Introduction

Cloud data storage provides a whole set of solutions to ease user access to data. Moreover, it supplies an organization with tangible competitive advantages: significant costs savings, improved degree of scalability, flexibility and resource pooling availability [4].

Deduplication refers to a technique for eliminating redundant files or more fine-grain blocks of data. Deduplication identifies common data blocks or files and only stores a single instance. By doing so, deduplication can drastically reduce the cloud space needed to store large datasets. However, as data security is becoming one of the most important requirements for cloud computing services, the need for secure deduplication has been greatly increased for specific security goals in diverse cloud application scenarios. Thus, the aim of secure deduplication is to provide both space efficiency and data security against both inside and outside adversaries [5].

Recently, He *et al*. [3], proposed a concrete scheme of deduplicatable dynamic PoS called DeyPoS, to achieve dynamic PoS (Proof of Storage) and secure cross-user deduplication, simultaneously. They also proved the security of the proposed scheme, and the theoretical analysis and experimental results showed that their construction is efficient in practice. However, after a close look at their proposed scheme, we identity some security weaknesses. First, there is compromise of data privacy. It is because confidentiality of DeyPoS scheme depends on secrecy of a hash value. However, a hash value of a message is not secret and thus an attacker obtained the hash value can compromise the data privacy. Also, if an attacker is a malicious cloud server, it can be done more easily. In this case, the malicious cloud server without fully storing the user's data can perform a forgery attack to deceive the user.

The rest of this paper is organized as follows. In section 2, we briefly describe DeyPoS scheme proposed in [3]. Then, in Section 3, we show the security vulnerability of DeyPoS scheme. Finally, we draw our conclusions in Section 4.

*Corresponding author: Sang Uk Shin, Dept. of IT Convergence and Application Engineering, Pukyong National University, 45, Yongso-ro, Nam-Gu. Busan, 48513, Republic of Korea, Tel: +81-51-629-6249

2 Brief Review of DeyPoS

He *et al*. proposed deduplicatable dynamic proof of storage called DeyPoS [3]. Figure 1 shows the system model of DeyPos scheme. This scheme achieves dynamic PoS and secure cross-user deduplication, simultaneously. DeyPoS considers two types of entities, the cloud server and users. The user is divided into an original user (first uploader) and a subsequent user (subsequent uploader). The proposed model assumes a malicious user and a malicious cloud server. The cloud server and users do not fully trust each other. There are five phases in DeyPoS system: pre-process, upload, deduplication, update, and proof of storage. They employ the following tools as our building blocks: Collision-resistant hash functions, $H(x)$, Deterministic symmetric encryption, $Enc_k(m)$, Hash-based message authentication codes, $HMAC_k(x)$, Pseudorandom functions, $f_k(x)$, Pseudorandom permutations, $\pi_k(x)$, and Key derivation functions, $KDF(k,x)$ (refer to [3] for details).

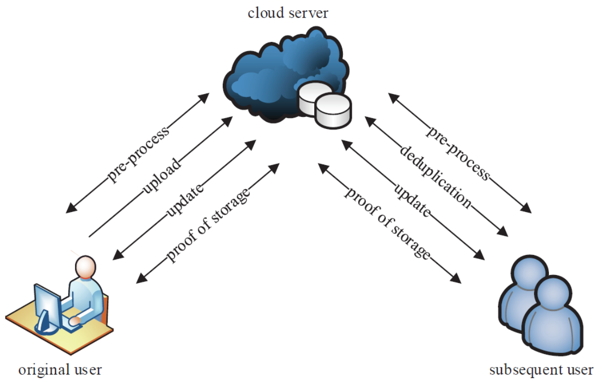


Figure 1: System model of Deypos

In the pre-process phase, a user computes the secret meta-data $e \leftarrow H(F)$ and a public id $id \leftarrow H(e)$ for a file F. And then the user announces that it has a certain file via id. In the upload phase, the user runs the encoding algorithm as follows. First, generate a random key $k \leftarrow \{0,1\}^{|e|}$, and compute $r \leftarrow k \oplus e$. Compute an encryption key $k_e \leftarrow KDF(k,0)$. For each file block $m_\iota (1 \leq \iota \leq n)$, compute $c_\iota \leftarrow Enc_{k_e}(m_\iota)$. Then the user builds a HAT (homomorphic authenticated tree) from F. Compute several parameters and tags of nodes in HAT, $\alpha_s \leftarrow KDF(k,1)$, $k_c \leftarrow KDF(k,2)$, $\alpha_c \leftarrow KDF(k,3)$, $\omega \leftarrow HMAC_e(t_1)$. Then set

$$C = \{c_1, c_2, \ldots, c_n\} \text{ and } \mathbf{T} = (r, \alpha_s, T, \omega, N).$$

where $T = \{\tau_1, \ldots, \tau_n\}$ and $N = \{v_1, v_2, \ldots\}$ is the set of all HAT nodes (refer to [3] for details). Here, the i-th node in HAT consists of $v_i = (i, l_i, v_i, t_i)$. An authentication tag t_i of the i-th node is computed by using pseudorandom function $f_{k_c}(i||l_i||v_i)$. At the end of the upload phase, the user uploads C and T to the cloud server and only stores e locally (refer to [3] for details).

If a file announced by a user in the pre-process phase exists in the cloud server, the user goes into the deduplication phase as follows. The server S first chooses a random $b \leftarrow [1,n]$ and $\kappa \leftarrow \{0,1\}^\lambda$. For each $j(1 \leq j \leq b)$, compute $\iota_j \leftarrow \pi_\kappa(b)$. Compute the path $\rho \leftarrow Path(I)$ and the siblings $\psi \leftarrow Sibling(\rho)$. Then send (r, b, κ, Q) to the user U. After receiving them from S, the user U computes $k \leftarrow r \oplus e$, $k_e \leftarrow KDF(k,0)$, $\alpha_s \leftarrow KDF(k,1)$, $k_c \leftarrow KDF(k,2)$, $\alpha_c \leftarrow KDF(k,3)$. For each file block $m_\iota (1 \leq \iota \leq n)$, $c_\iota \leftarrow Enc_{k_e}(m_\iota)$. For each $j(1 \leq j \leq b)$, compute $\iota_j \leftarrow \pi_\kappa(b)$. And run the deduplication proving algorithm with S.

At any time, users can go into the proof of storage phase if they have the ownerships of the files. The users and the cloud server run the checking protocol as follows. The user U chooses $b \leftarrow [1,n]$

and $\kappa \leftarrow \{0,1\}^{\lambda}$, and sends (b,κ) to the server S. For each $j(1 \leq j \leq b)$, S computes $\iota_j \leftarrow \pi_\kappa(b)$ and then invokes the response algorithm. S sends the proof res to U with (r,ν_1,ω). U computes $k \leftarrow r \oplus e$, $\alpha_s \leftarrow KDF(k,1)$, $k_c \leftarrow KDF(k,2)$, $\alpha_c \leftarrow KDF(k,3)$. Then, it verifies ν_1 and invokes the verification algorithm. It outputs 1 if the verification succeeds and 0 otherwise.

3 Analysis of DeyPoS

In this section, we discuss the vulnerabilities in DeyPoS scheme [3].

3.1 Security Models of DeyPoS

In [3], authors assumed that the files to be used in DeyPoS scheme are unpredictable which means the files have high min-entropy so that an adversary cannot evaluate any meaningful information by guessing a file. The assumption is valid in the single-user setting since each user uses a random private key to encrypt his files stored in the server's storage. In this case, a different ciphertext is generated for each user due to the use of a different encryption key, and thus it is reasonable to assume the unpredictability of stored files. However, in the multi-user setting, the same ciphertext should be made for the same file to support deduplication. Hence, it is not reasonable to assume the unpredictability of a file since the deduplication of encrypted data in multi-user setting is one of main services of DeyPoS.

The security analysis of DeyPoS is correct when we assume the unpredictability of files, but the assumption is not reasonable as we explained in the above. Here we want to emphasis that the multi-user setting should be considered for correct security analysis of DeyPoS. In the setting, we can assume the adversary who tries to break the security of DeyPoS by performing some kinds of guessing attacks since the security of the scheme can be damaged if an adversary can guess stored files. In the following sections, we will show that the security of DeyPoS cannot be guaranteed if we analyze its security in the multi-user setting.

3.2 Compromise of Data Privacy

If we analyze the security of DeyPoS in the multi-user setting, we can not expect sufficient data privacy against guessing attacks. Recall that, to perform the duplicate check, DeyPoS scheme uses $id \leftarrow H(e)$ as the file identifier, where the secret meta-data e is derived from $e \leftarrow H(F)$. Thus, it is possible for an attacker to perform brute-force attack for a predictable message set. An attacker computes id for all messages in predictable message space, checks whether the deduplication phase is executed. This allows the attacker to obtain the secret meta-data e. Note that the above attack strategy is not valid in the single-user setting, but we have to consider the multi-user setting as we explained in Section 3.1.

Moreover, a hash value of a message is not secret in practice even though DeyPoS scheme assumes the unpredictable message space. It is well-known that the hash value of a message is easily obtainable in many ways [1] without performing the above described brute-force attack. Hence, the confidentiality of DeyPoS scheme can be easily damaged since the security feature entirely depends on the secrecy of a hash value. If the attacker has obtained the hash value of the message, he can compute $id \leftarrow H(e)$ and then check if the message is duplicated on the cloud server. Thus the data privacy can be compromise through attacks such as identifying files and learning the contents of files [2].

3.3 Compromise of Unforgeability

In [2], DeyPoS is claimed to be secure against forgery attack. However, unlike authors' argument, it may be possible to perform a forgery attack to deceive the user in the proof of storage phase. DeyPoS scheme uses HAT (Homomorphic Authenticated Tree) to support integrity verification, dynamic operations, and cross-user deduplication. In HAT an authentication tag is computed by using pseudorandom function $f_{k_c}(x)$. An input value x consists of $(i||l_i||v_i)$, where i is the index of the node, l_i is the number of leaf nodes that can be reached from the i-th node and v_i is the version number of the i-th node. Here, we want to remind that the security of pseudorandom function $f_{k_c}(x)$ entirely depends on the secret key k_c.

Recall that the masked secret key r is stored in the cloud server. Hence, a malicious cloud server can obtain the key k easily by performing brute-force attack to guess e and computing $r \oplus e$. Using the key k, the malicious cloud server can derive $k_c \leftarrow KDF(k,2)$. Thus, the malicious cloud server can manipulate a proof *res* in order to pass the verification algorithm in the user side. This is because the user with no other data except for the secret value e verifies the proof *res* from the server. Here the authentication tag t_i is derived from the pseudorandom function $f_{k_c}(x)$ depending on the secret key k_c. Thus, the malicious cloud server that has obtained k_c without fully storing the user's data can perform a forgery attack to deceive the user.

3.4 Lack of Undeniability

Generally, for storage-based services, the service provider is regarded as an honest but curious adversary who could behave maliciously if his behave does not detected by others and he obtains any merit from his malicious behavior. In [3], authors assumed little-bit stronger adversary who can succeed in attack regardless of being detected or not. However, DeyPoS is not secure against an honest but curious adversary who is weaker than the adversary considered in [3]. Due to the lack of undeniability of DeyPoS regarding the file update, the server can ignore any update request given by users if the request causes the increase of the volume of stored file. The user can recognize that the stored file is not up-to-date when he performs the proof of storage protocol, but the server can deny about it since the client doesn't have any undeniable evidence for the update. Though the user has metadata for the latest version, it doesn't means that the version was stored in the storage server. In this scenarios, the adversarial server can reduce the cost of storage without being detected by users. Though an honest server does not intentionally try to fake his clients, he feels free from the responsibility of maintaining the latest version along with the users' request.

4 Conclusions

In this paper, we have identified the vulnerabilities in DeyPoS scheme for supporting dynamic PoS and secure cross-user deduplication, simultaneously. Confidentiality of DeyPoS scheme depends on secrecy of a hash value. However, a hash value of a message is not secret and thus an attacker can compromise the data privacy. Also, if an attacker is a malicious cloud server, it can be done more easily. In this case, the malicious cloud server without fully storing the user's data can perform a forgery attack to deceive the user. Moreover, one of merits of DeyPoS can be harmed since the update cannot be guaranteed in the scheme.

Acknowledgments

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [19ZH1200, Development of core technologies for trust data connectome to realize a safe data society] and by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program IITP-2019-2015-0-00403)supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

References

[1] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg. Proofs of ownership in remote storage systems. In *Proc. of the 18th ACM conference on Computer and Communications Security (CCS'11), Chicago, Illinois, USA*, pages 491–500. ACM Press, October 2011.

[2] D. Harnik, B. Pinkas, and A. Shulman-Peleg. Side channels in cloud services: Deduplication in cloud storage. *IEEE Security & Privacy*, 8(6):40–47, Novemver-December 2010.

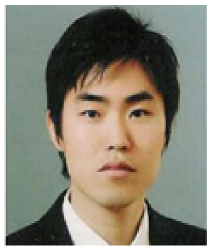
[3] K. He, J. Chen, R. Du, Q. Wu, G. Xue, and X. Zhang. Deypos: Deduplicatable dynamic proof of storage for multi-user environments. *IEEE Transactions on Computers*, 65(12):3631–3645, December 2016.

[4] S. Islam, M. Ouedraogo, C. Kalloniatis, H. Mouratidis, and S. Gritzalis. Assurance of security and privacy requirements for cloud deployment models. *IEEE Transactions on Cloud Computing*, 6(2):387–400, April-June 2018.

[5] Y. Shin, D. Koo, and J. Hur. A survey of secure data deduplication schemes for cloud storage systems. *ACM Computing Surveys*, 49(4):3–8, February 2017.

Author Biography

Taek-Young Youn received his BS, MS, and Ph.D from Korea University in 2003, 2005, and 2009, respectively. He is currently a senior researcher at Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests include cryptography, information security, and data privacy.



Nam-Su Jho received the B.S. degree in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1999 and the Ph.D. degree in mathematics from Seoul National University, Seoul, Korea, in 2007. Since 2007, he has been with the Electronics and Telecommunications Research Institute (ETRI) as a senior researcher. His research interests include cryptography and information theory.

Si-Wan Noh received his M.S. degree in Interdisciplinary Program of Information Security from Pukyong University, Republic of Korea in 2018. He is currently a doctoral course student of Pukyong National University. His research interests are related with blockchain, applied cryptography, and communication security.

Kyung-Hyune Rhee received his M.S. and Ph.D. degrees from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea in 1985 and 1992, respectively. He worked as a senior researcher in Electronic and Telecommunications Research Institute (ETRI), Daejeon, Korea from 1985 to 1993. He also worked as a visiting scholar in the University of Adelaide in Australia, the University of Tokyo in Japan, the University of California at Irvine in USA, and Kyushu University in Japan. He has served as a Chairman of Division of Information and Communication Technology, Colombo Plan Staff College for Technician Education in Manila, the Philippines. He is currently a professor in the Department of IT Convergence and Application Engineering, Pukyong National University, Busan, Korea. His research interests center on multimedia security and analysis, key management protocols and mobile ad-hoc and VANET communication security.

Sang Uk Shin received his M.S. and Ph.D. degrees from Pukyong National University, Busan, Korea in 1997 and 2000, respectively. He worked as a senior researcher in Electronics and Tele-communications Research Institute, Daejeon Korea from 2000 to 2003. He is currently a professor in Department of IT Convergence and Application Engineering, Pukyong National University. His research interests include digital forensics, e-Discovery, cryptographic protocol, mobile and wireless network security and multimedia content security.