Journal of Southeast University (English Edition) Vol. 28, No. 4, pp. 392 - 397 Dec. 2012 ISSN 1003—7985

Security analysis and improvement on resilient storage outsourcing schemes in mobile cloud computing

Liu Xiao Jiang Rui

(School of Information Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: The resilient storage outsourcing schemes in mobile cloud computing are analyzed. It is pointed out that the sharing-based scheme (ShS) has vulnerabilities regarding confidentiality and integrity; meanwhile, the coding-based scheme (CoS) and the encryption-based scheme (EnS) have vulnerabilities on integrity. The corresponding attacks on these vulnerabilities are given. Then, the improved protocols such as the secure sharing-based protocol (SShP), the secure coding-based protocol (SCoP) and the secure encryption-based protocol (SEnP), are proposed to overcome these vulnerabilities. The core elements are protected through public key encryptions and digital signatures. Security analyses show that the confidentiality and the integrity of the improved protocols are guaranteed. Meanwhile, the improved protocols can keep the frame of the former schemes and have higher security. The simulation results illustrate that compared with the existing protocols, the communication overhead of the improved protocols is not significantly increased.

Key words: mobile cloud computing; cloud storage; security protocols

doi: 10.3969/j.issn.1003-7985.2012.04.004

Mobile cloud computing provides powerful capabilities of data processing and data storage, leading to the data migration from mobile devices to the cloud. Since cloud computing provides various resources necessary for computing, different security technologies are required for each type of resource[1]. Mobile cloud computing combines the advantages of both the mobile technologies and the cloud computing technologies, facing the security challenges from both the mobile devices (MDs) and the cloud.

Received 2012-05-04.

Biographies: Liu Xiao (1989—), male, graduate; Jiang Rui (corresponding author), male, doctor, associate professor, R. Jiang@seu.edu.cn.

Foundation items: The National Natural Science Foundation of China (No. 60902008), the Key Laboratory Hi-Tech Program of Changzhou City (No. CM20103003), the Key Laboratory Program of Information Network Security of Ministry of Public Security (No. C12602), the Science and Technology Supporting Project of Changzhou City (No. CE20120030).

Citation: Liu Xiao, Jiang Rui. Security analysis and improvement on resilient storage outsourcing schemes in mobile cloud computing. [J]. Journal of Southeast University (English Edition), 2012, 28(4): 392 - 397. [doi: 10.3969/j.issn.1003-7985.2012.04.004]

With the development of cloud computing, data are gradually migrated from mobile devices to the cloud. Secure storage becomes users' greatest concern. A number of secure cloud storage mechanisms have been studied[2-5]. Hsueh et al.[6] proposed a mechanism that integrates cloud storage, hybrid cryptography, and digital signatures to provide security requirements for data storage of mobile phones. Ruiz-Alvarez et al.[7] presented an automatic approach to selecting the cloud storage service that best matches each dataset of a given application. Feng et al.[8] analyzed the integrity vulnerability existing in the current cloud storage platforms and proposed a novel non-repudiation protocol which is specifically designed in the context of cloud computing environments. Unfortunately, few of them can successfully guarantee the confidentiality and the integrity of users' data during storage.

To guarantee the confidentiality and the integrity of users' data during uploading and downloading, Ren et al.[9] proposed a family of schemes which include the sharing-based scheme (ShS), the coding-based scheme (CoS) and the encryption-based scheme (EnS) for different situations. The authors declared that those schemes managed to guarantee the confidentiality and the integrity of users' data. However, by our analysis, the ShS can guarantee neither the confidentiality nor the integrity of users' data; besides, the CoS and the EnS fail to guarantee the integrity of users' data.

In this paper, we focus on the secure data migration from mobile devices to cloud servers (CSs). First, we point out the vulnerabilities existing in the resilient storage outsourcing schemes and present the corresponding attacks. The ShS can guarantee neither the confidentiality nor the integrity of users' data, which will cause the failure of data download and the disclosure of users' data. The CoS and the EnS fail to guarantee the integrity of users' data, which will cause the failure of data download. To overcome these security shortages, we make improvements on the resilient storage outsourcing schemes and propose three secure storage outsourcing protocols. They are the secure sharing-based protocol (SShP), the secure coding-based protocol (SCoP) and the secure encryption-based protocol (SEnP). Our protocols can guarantee both the confidentiality and the integrity of users' data during uploading and storage. The security analyses

demonstrate that the proposed protocols can realize secure storage outsourcing in mobile cloud computing and resist against various attacks. The simulation results show that our improvement has little influence on communication overhead.

1 Brief Review of ShS, EnS, CoS and Attacks

In this section, we briefly introduce the resilient storage outsourcing schemes such as ShS, CoS and EnS proposed by Ren et al[9].

1.1 Sharing-based scheme (ShS)

The ShS is composed of the uploading process and the downloading process. This scheme needs multiple CSs (see Fig. 1).

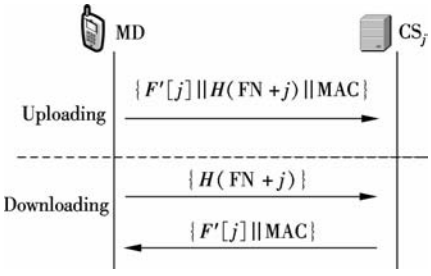

Fig. 1 Sharing based-scheme (ShS)

At the uploading phase, users input password PWD from mobile device (MD). MD computes the integrity key $IK = H(FN \,||\, PWD \,||\, FS)$ and the file integrity authentication code $MAC = H(F, IK)$. Here $H()$ represents the Hash function; FN is the file name and FS is the file size. Then MD generates $d-1$ files $F'[i]$ $(1 \leqslant i \leqslant d-1)$ randomly and obtains the final file $F'[d]$ by $F'[d] = \oplus_{i=1}^{d-1} F'[i] \oplus F$ where $|F'[i]| = |F|$. After that, MD sends the message $\{F'[j] \,||\, H(FN+j) \,||\, MAC\}$ to CS_j $(1 \leqslant j \leqslant d)$ and stores FN locally.

At the downloading phase, MD sends $H(FN+j)$ to CS_j. Then CS_j searches the file $F'[j]$ that matches $H(FN+j)$ and sends back the corresponding message $\{F'[j] \,||\, MAC\}$ to MD. MD can recover F by $F = \oplus_{i=1}^{d} F'[i]$.

1.2 Coding-based scheme (CoS)

The CoS is also composed of the uploading process and the downloading process. This scheme still needs multiple CSs. The whole process of the CoS is similar to that of the ShS as shown in Fig. 1. Suppose that there exists d portal CSs.

At the uploading phase, MD divides F into $|F|/n$ parts ($dt = |F|/n$), where n is the length of the hash value. Each part has n bits of data and it is marked as $F[i][j]$, $1 \leqslant i \leqslant t$, $1 \leqslant j \leqslant d$. Users input password PWD from MD. MD computes $\alpha_i = H^i(PWD \,||\, FN \,||\, FS)$, the integrity key $IK = H(\alpha_1 \,||\, \ldots \,||\, \alpha_t)$, the file integrity authentication code $MAC = H(F, IK)$ and the coding vector $\hat{\boldsymbol{\alpha}} = \{\alpha_1, \ldots, \alpha_t\}$, where $H^i()$ is the Hash function with i times iterative operations. Then MD computes $F'[j] = \sum_{i=1}^{t} \alpha_i F[i][j]$ and sends the message $\{F'[j] \,||\, H(FN+j) \,||\, MAC\}$ to CS_j, $1 \leqslant j \leqslant d$. Meanwhile, MD stores FN locally.

At the downloading phase, MD sends $H(FN+j)$ to CS_j. Then CS_j searches the file $F'[j]$ that matches $H(FN+j)$ and sends back the corresponding message $\{F'[j] \,||\, MAC\}$ to MD. MD can recover F by $F[i][j] = \hat{\boldsymbol{\alpha}}^{-1}[i] F'[j]$.

1.3 Encryption-based scheme (EnS)

The EnS is also composed of the uploading process and the downloading process. This scheme is applied to the situations where only one CS is presented. The whole process of the EnS is similar to that of the ShS as shown in Fig. 1.

At the uploading phase, users input password PWD from MD. MD computes the encryption key $EK = H(PWD \,||\, FN \,||\, FS)$, the integrity key $IK = H(FN \,||\, PWD \,||\, FS)$, the encrypted file $F' = \{F\}_{EK}$ and the file integrity authentication code $MAC = H(F, IK)$, where $\{\}_K$ is the symmetric key encryption with key K. Then MD sends the message $\{F' \,||\, H(FN) \,||\, MAC\}$ to CS and stores FN locally.

At the downloading phase, MD sends $H(FN)$ to CS. Then CS searches the file F' that matches $H(FN)$ and sends back the corresponding message $\{F' \,||\, MAC\}$ to MD. MD can recover F by $F = \{F'\}_{EK^{-1}}$ where $\{\}_{K^{-1}}$ is the symmetric key decryption with key K.

1.4 Attacks

In the scheme ShS, the authors declare that during uploading and storage, the confidentiality and the integrity of the file are guaranteed. However, this security goal is realized under the strong assumption that the link security is guaranteed. In fact, this assumption is merely an ideal situation in the wireless environment. In the ShS, only one message is transferred during the uploading process and there lacks file verification in CS_j. Hence, it is impossible for MD to figure out whether CS_j has stored the data successfully or not and whether the file is modified. Therefore, it is easy for an attacker to modify the user's data during the uploading process and the user cannot be aware of this attack until he downloads the whole file. Moreover, the ShS has to face fatal problems. Through passive eavesdropping, it is easy for an attacker to capture all the files sent to CSs by monitoring the link between MD and CSs. Then the attacker can recover the original file by adding up all the files. These vulnerabilities make it impossible to protect the confidentiality and the integrity of users' data during uploading and storage.

In the ShS, the detail of the man-in-the-middle attack is shown in Fig. 2. In the first step of the ShS, the attacker first captures the message to be transferred to CS_j

through monitoring the link between MD and CS_j. Then he modifies the file $F'[j]$ and creates a new one $F''[j]$ to replace $F'[j]$. After that, the attacker transfers the modified message $\{F''[j] \,||\, H(\mathrm{FN}+j) \,||\, \mathrm{MAC}\}$ to CS_j. However, the CS_j cannot discover that the message is modified because there lacks file verification of $F'[j]$ in CS_j.

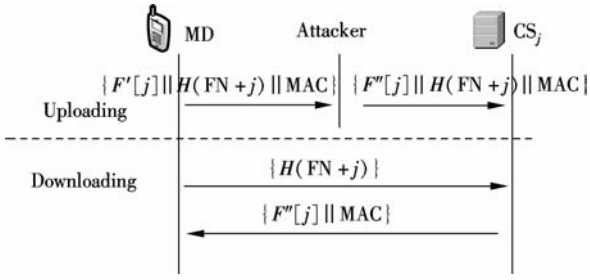

Fig. 2 Man-in-the-middle attack on ShS

In the ShS, the detail of the passive eavesdropping attack is shown in Fig. 3. At the first step of the ShS, through monitoring the data stream between MD and CSs, the attacker can capture all $F'[i], 1 \leqslant i \leqslant d$. Hence, he can illegally recover F by adding up all $F'[i]$ through XOR operations $F = \oplus_{i=1}^{d} F'[i]$ in the end.

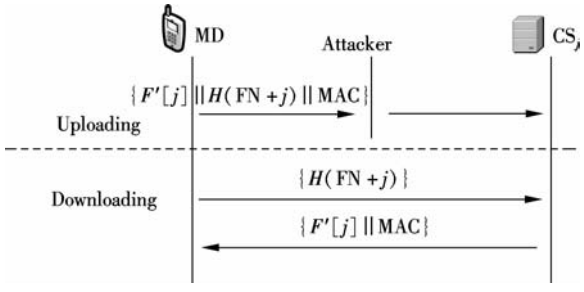

Fig. 3 Passive eavesdropping attack

The man-in-the-middle attack in the CoS and the EnS is similar to that in the ShS as shown in Fig. 3.

2 Improvement on Resilient Storage Outsourcing Schemes

In this section, we introduce SShP, SCoP and SEnP to overcome the security vulnerabilities existing in resilient storage outsourcing schemes.

2.1 Secure sharing-based protocol (SShP)

Our improved protocol SShP is composed of the uploading process and the downloading process. In our protocol, we introduce a file A generated by MD to participate in the XOR operations. In this way, the confidentiality of users' data can be guaranteed. Meanwhile, we adopt digital signature and public key encryption to guarantee the integrity of users' data. Moreover, at the uploading phase, we add a message sent back from CSs to MD. In this way MD can confirm that CSs have stored users' data successfully. This protocol is applied to situations where there exist multiple CSs. The detailed process of the SShP is shown in Fig. 4.

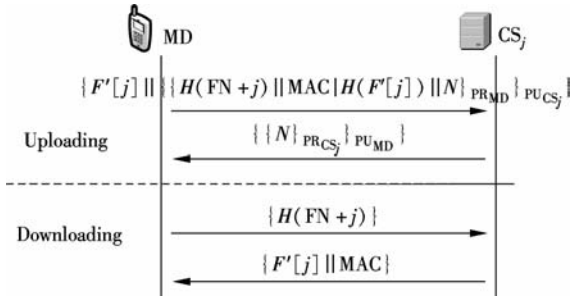

Fig. 4 Secure sharing-based protocol (SShP)

2.1.1 Uploading process of SShP

1) Users input password PWD from MD. MD computes the integrity key $\mathrm{IK} = H(\mathrm{FN} \,||\, \mathrm{PWD} \,||\, \mathrm{FS})$ which is used to verify the integrity of the user's file. Meanwhile, MD calculates parameters α_i by $\alpha_i = H^i(\mathrm{PWD} \,||\, \mathrm{FN} \,||\, \mathrm{FS})\,(1 \leqslant i \leqslant t)$, where $(t-1)n < |F| < tn$ and n is the length of the hash value. Then MD composes file A' by $A' = (\alpha_1 \,||\, \alpha_2 \ldots \,||\, \alpha_t)$. After that MD takes the previous $|F|$ bits value of A' and we denote this file as A. Now, A has the same length as F.

2) MD generates $d-1$ files $F'[i]$ $(1 \leqslant i \leqslant d-1)$ randomly, where $|F'[i]| = |F| = \mathrm{FS}$. MD computes $F'[d] = \oplus_{i=1}^{d-1} F'[i] \oplus F \oplus A$. Files $F'[i]\,(1 \leqslant i \leqslant d)$ will be transferred to different CS_j separately. Then MD calculates the hash values of these files, $H(F'[j])\,(1 \leqslant j \leqslant d)$ and the hash value of each FN $+ j$, $H(\mathrm{FN}+j)\,(1 \leqslant j \leqslant d)$.

3) MD chooses a random number N to keep freshness.

4) MD generates the digital signature $\{H(\mathrm{FN}+j) \,||\, \mathrm{MAC} \,||\, H(F'[j]) \,||\, N\}_{\mathrm{PR}_{\mathrm{MD}}}$, where $\{\}_{\mathrm{PR}_A}$ is a digital signature with the private key of A. Then MD encrypts this signature with the public key of each CS_j.

5) MD sends the message $\{F'[j] \,||\, \{\{H(\mathrm{FN}+j) \,||\, \mathrm{MAC} \,||\, H(F'[j]) \,||\, N)\}_{\mathrm{PR}_{\mathrm{MD}}}\}_{\mathrm{PU}_{\mathrm{CS}_j}}\}$ to $CS_j\,(1 \leqslant j \leqslant d)$, where $\{\}_{\mathrm{PU}_A}$ is the public key encryption with the public key of A.

6) When CS_j has received the message $\{F'[j] \,||\, \{\{H(\mathrm{FN}+j) \,||\, \mathrm{MAC} \,||\, H(F'[j]) \,||\, N)\}_{\mathrm{PR}_{\mathrm{MD}}}\}_{\mathrm{PU}_{\mathrm{CS}_j}}\}$ from MD, he decrypts this message and obtains the value $H(\mathrm{FN}+j)$, MAC, $H(F'[j])$ and the random number N. Next CS_j verifies the integrity of $F'[j]$ by computing the hash value of $F'[j]$. If the hash value he computes equals the hash value revealed by decryption, CS_j stores $F'[j]$, $H(\mathrm{FN}+j)$, MAC.

7) CS_j generates the digital signature $\{N\}_{\mathrm{PR}_{\mathrm{CS}_j}}$ and encrypts this digital signature with the public key of MD. Then CS_j responses to MD by sending the message $\{\{N\}_{\mathrm{PR}_{\mathrm{CS}_j}}\}_{\mathrm{PU}_{\mathrm{MD}}}$ to MD.

8) When MD has received the message $\{\{N\}_{\mathrm{PR}_{\mathrm{CS}_j}}\}_{\mathrm{PU}_{\mathrm{MD}}}$, he compares whether this N equals the one transferred to CS_j before.

2.1.2 Downloading process of SShP

1) When MD wants to download file F, he sends $H(\mathrm{FN}+j)$ to $CS_j\,(1 \leqslant j \leqslant d)$.

2) CS_j searches the file that matches $H(FN+j)$ and sends back the corresponding message $\{F'[j] \,||\, MAC\}$ to MD.

3) When MD has received all $F'[j]\ (1 \leqslant j \leqslant d)$, he asks the user to input the password PWD through MD. MD computes the integrity key $IK = H(FN \,||\, PWD \,||\, FS)$. Meanwhile, MD calculates parameters α_i by $\alpha_i = H^i(PWD \,||\, FN \,||\, FS)\ (1 \leqslant i \leqslant t)$. Then MD composes file A' by $A' = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$. After that MD takes the previous $|F|$ bits value of A' and generates the file A.

4) MD recovers the user's file by $F = \oplus_{i=1}^{d} F'[i] \oplus A$. After that MD verifies the integrity of F by calculating $MAC = H(F, IK)$. Then MD compares whether this MAC value equals the MAC value contained in the message received.

2.2 Secure coding-based protocol (SCoP)

Our improved protocol SCoP is composed of the uploading process and the downloading process. The SCoP is also applied to the situations where exists multiple CSs. The structure of the SCoP is the same as that of the SShP as shown in Fig. 4.

2.2.1 Uploading process of SCoP

1) Suppose that there exists d CSs. MD divides F into $|F|/n$ parts ($d = |F|/(nt)$). Each part has n bits data and it can be marked as $F[i][j], 1 \leqslant j \leqslant d, 1 \leqslant i \leqslant t$.

2) The user inputs password PWD from MD. MD computes parameters α_i by $\alpha_i = H^i(PWD \,||\, FN \,||\, FS)$. Then MD generates the integrity key $IK = (\alpha_1 \,||\, \ldots \,||\, \alpha_t)$ and the coding vector $\hat{\boldsymbol{\alpha}} = \{\alpha_1, \ldots, \alpha_t\}$. After that, MD calculates the file integrity authentication code $MAC = H(F, IK)$.

3) MD generates coded file by $F'[j] = \sum_{i=1}^{t} \alpha_i F[i][j]$ $(1 \leqslant j \leqslant d)$.

4) MD computes each $H(F'[j])$ and $H(FN+j)$. Meanwhile, MD chooses a random number N to keep freshness.

5) MD generates the digital signature $\{H(FN+j) \,||\, MAC \,||\, H(F'[j]) \,||\, N\}_{PR_{MD}}$ and encrypts this digital signature with the public key of CS_j.

6) MD sends the message $\{F'[j] \,||\, \{\{H(FN+j) \,||\, MAC \,||\, H(F'[j]) \,||\, N\}_{PR_{MD}}\}_{PU_{CS_j}}\}$ to each $CS_j\ (1 \leqslant j \leqslant d)$ separately.

7) When CS_j has received the message $\{F'[j] \,||\, \{\{H(FN+j) \,||\, MAC \,||\, H(F'[j]) \,||\, N\}_{PR_{MD}}\}_{PU_{CS_j}}\}$ from MD, he decrypts this message and computes $H(F'[j])$ himself. If the hash value he computes equals the hash value revealed by decryption, CS_j stores $H(F'[j])$, $H(FN+j)$, MAC. Then CS_j sends back the message $\{\{N\}_{PR_{CS_j}}\}_{PU_{MD}}$ to MD.

8) When MD has received the message $\{\{N\}_{PR_{CS_j}}\}_{PU_{MD}}$, he decrypts it and compares whether N equals the random number transferred to CS_j before.

2.2.2 Downloading process of SCoP

1) When MD wants to download file F, he sends $H(FN+j)$ to each CS_j.

2) CS_j searches the file that matches $H(FN+j)$ and sends back the corresponding message $\{F'[j] \,||\, MAC\}$ to MD.

3) When MD has received all $F'[j]\ (1 \leqslant j \leqslant d)$ from CS_j, he asks the user to input the password PWD through MD. MD computes parameters α_i by $\alpha_i = H^i(PWD \,||\, FN \,||\, FS)$. Then MD can recover the file F by $F[i][j] = \hat{\boldsymbol{\alpha}}^{-1}[i] F'[j]$.

4) MD computes the corresponding IK and MAC.

5) MD checks the integrity of F by comparing the MAC.

2.3 Secure encryption-based protocol (SEnP)

Our improved protocol SEnP is applied to the situations where only one CS is available. The detailed process of the SEnP is shown in Fig. 5.

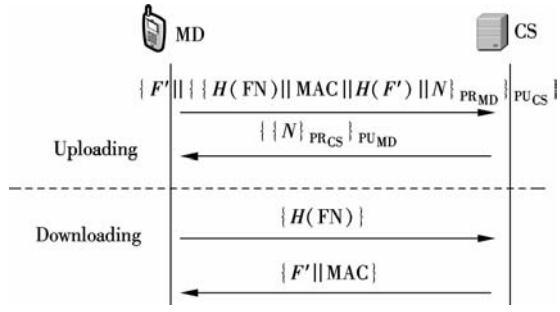

Fig. 5 Secure encryption-based protocol (SEnP)

2.3.1 Uploading process of SEnP

1) The user inputs password PWD from MD. Then MD computes the encryption key $EK = H(PWD \,||\, FN \,||\, FS)$ and the integrity key $IK = H(FN \,||\, PWD \,||\, FS)$.

2) MD encrypts file F with EK. Then MD gets the encrypted file $F' = \{F\}_{EK}$, where $|F'| = |F| = FS$. After that, MD calculates $MAC = H(F, IK)$ and $H(F')$. Meanwhile, MD chooses a random number N to keep freshness.

3) MD generates the digital signature $\{H(FN) \,||\, MAC \,||\, H(F') \,||\, N\}_{PR_{MD}}$ and encrypts this digital signature with the public key of CS.

4) MD sends the message $\{F' \,||\, \{\{H(FN) \,||\, MAC \,||\, H(F') \,||\, N\}_{PR_{MD}}\}_{PU_{CS}}\}$ to CS.

5) When CS has received the message $\{F' \,||\, \{\{H(FN) \,||\, MAC \,||\, H(F') \,||\, N\}_{PR_{MD}}\}_{PU_{CS}}\}$ from MD, he decrypts the message and computes $H(F')$. If the hash value he computes equals the hash value calculated by decryption, CS stores F', $H(FN)$ and MAC. Then CS sends back $\{\{N\}_{PR_{CS}}\}_{PU_{MD}}$ to MD.

6) When MD has received the message $\{\{N\}_{PR_{CS}}\}_{PU_{MD}}$, he decrypts it and compares whether N equals the random number transferred to CS.

2.3.2 Downloading process of SEnP

1) MD sends $H(FN)$ to CS.

2) CS searches the file that matches $H(FN)$ and sends back the corresponding F' and MAC to MD.

3) When MD has received the message from CS, he asks the user to input the password PWD. Then MD computes the encryption key EK and the integrity key IK.

4) MD recovers the file by $F = \{F'\}_{EK^{-1}}$. Afterwards MD computes MAC and checks whether it equals the MAC value he has received.

3 Security Analysis

3.1 Security analysis of SShP

3.1.1 Confidentiality

In the original ShS, the user's file can be illegally recovered by adding up all the files stored in the CSs through XOR operations. Therefore, the ShS fails to guarantee the confidentiality.

In our improved SShP, we introduce a file A created by a certain amount of the hash operations. In order to reveal F, MD has to recalculate A. Meanwhile, MD has to add up the files stored in CSs and file A. However, file A is related to file name FN, file size FS and password PWD. So the attacker cannot recover F because he cannot calculate A even if he obtains all the messages during transmission. Moreover, the confidentiality of $H(\mathrm{FN} + j)$, MAC, $H(F'[j])$ and N can be guaranteed because they are encrypted by the public key of CS_j. Similarly, the confidentiality of N in $\{\{N\}_{\mathrm{PR_{CS}}}\}_{\mathrm{PU_{MD}}}$ is guaranteed since the message is encrypted by the public key of MD.

3.1.2 Integrity

Through our analysis, the original ShS cannot guarantee the integrity of any piece of the file transferred to each CS_j during uploading and storage. In the ShS, an attacker can change the value of $F'[j]$ during uploading. However MD cannot be aware of the change of $F'[j]$ until he downloads the whole file. Only when he downloads the whole file, can he figure out the modification of F. It will take MD much time and communication overhead to realize this modification.

In SShP, we introduce the hash value of $F'[j]$ which is sent to CS_j. In this way, CS_j can check the integrity of $F'[j]$ and response to MD whether the file has been modified. Therefore, the man-in-the-middle attack does not exist in the SShP and the integrity of F during uploading and storage can be guaranteed.

3.1.3 Non-repudiation

In the original ShS, the digital signature technique is not adopted. Thus, the ShS cannot provide non-repudiation. Therefore, the fabrication attack can be realized. Moreover, vicious CSs may repudiate the reception of the file which will cause the failure of the download.

In our improved SShP, we adopt the signature technique to protect our data. The hash value of $F'[j]$ and the random number N are included in the signature $\{(H(\mathrm{FN}+j) \,||\, \mathrm{MAC} \,||\, H(F'[j]) \,||\, N\}_{\mathrm{PR_{MD}}}$; therefore, CS_j can confirm that the message is sent by MD instead of an attacker. Similarly, the random number in the message sent back is included in the signature $\{N\}_{\mathrm{PR}}$; therefore, MD can confirm that the message sent to CS_j has been stored successfully.

3.1.4 Avoiding the man-in-the-middle attack

Through our analysis, the original ShS cannot resist the man-in-the-middle attack. An attack can modify the data to be stored in CS_j. This vulnerability will cause serious problems.

In our improved SShP, the hash value of $F'[j]$ can guarantee that $F'[j]$ has not been modified because the hash function is an one-way function, so the attacker cannot reveal $F'[j]$ from $H(F'[j])$. The encryption of the hash value guarantees that the hash value $H(F'[j])$ cannot be obtained by the attacker because the attacker cannot obtain the private key of the CSs. So the attacker cannot decrypt the message and reveal the hash value. Moreover, the digital signature $\{H(\mathrm{FN}+j) \,||\, \mathrm{MAC} \,||\, H(F'[j]) \,||\, N\}_{\mathrm{PR_{MD}}}$ guarantees that the data the CSs have received is sent by the user instead of the attacker.

3.2 Security analysis of SCoP and SEnP

The structure of the SCoP and the SEnP is similar to that of the SShP; hence, it is similar to certify that the integrity, non-repudiation, avoiding the man-in-the middle attack of the SCoP and the SEnP can be realized. Therefore, we mainly discuss the confidentiality of the SCoP and the SEnP in this part.

In the SCoP, the confidentiality of the user's data can be guaranteed because the file to be stored in CS_j is coded with the coding vector $\hat{\alpha}$. In order to distinguish F from $F'[j]$, the attacker has to know the coding vector $\hat{\alpha}$. However it is impossible for the attacker himself to calculate $\hat{\alpha}$ since the password PWD and file name FN are not delivered.

In the SEnP, the confidentiality of the user's data can be guaranteed because the file to be stored in CS is encrypted with encryption key EK. In order to reveal F from F', the attacker has to know the encryption key EK. However, it is impossible for the attacker himself to calculate EK since the password PWD and file name FN are not delivered.

4 Performance Simulation

We make the performance simulation of the resilient storage outsourcing protocols and our improved protocols with NS2. We assume that the bandwidth of the wireless network is 3 Mbit/s and the size of user's file is 3 MB. In the ShS, SShP, CoS and SCoP, we assume that there exist three CSs. In the simulation, we adopt the data encryption algorithm (DES) as our encryption algorithm. The time delay and the throughput of these protocols during the uploading process are shown in Fig. 6 and Fig. 7. The performance simulation illustrates that our improvement has little influence on the communication overhead.

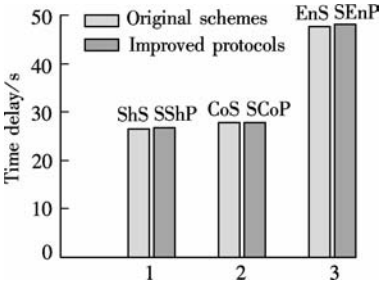

Fig. 6 Time delay during uploading process

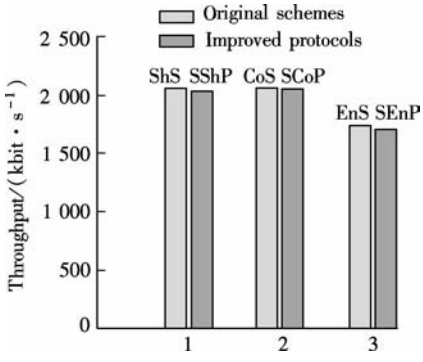

Fig. 7 Throughput during uploading

5 Conclusion

In this paper, we review a family of storage outsourcing schemes in mobile cloud computing. Through our analysis, we point out that those schemes have fatal vulnerabilities and we present the corresponding attacks. Finally, to overcome these vulnerabilities, we propose a family of improved schemes SShP, SCoP and SEnP, which can retain the merits of the resilient storage outsourcing schemes and overcome their security shortages.

References

[1] Park Ji Soo, Yi Ki Jung, Park Jong Hyuk. SSP-MCloud: a study on security service protocol for smartphone centric mobile cloud computing [C]//*Lecture Notes in Electrical Engineering*. Springer, 2012, **107**: 165 – 172.

[2] Feng Jun, Chen Yu, Summerville D, et al. Enhancing cloud storage security against roll-back attacks with a new fair multi-party non-repudiation protocol [C]//2011 *IEEE Consumer Communications and Networking Conference*. New York, USA, 2011: 521 – 522.

[3] Bermbach D. Meta storage: a federated cloud storage system to manage consistency-latency tradeoffs [C]//2011 *IEEE 4th International Conference on Cloud Computing*. Washington DC, USA, 2011: 452 – 459.

[4] Zhang Xinwen, Joshua Schiffman, Simon Gibbs, et al. Securing elastic applications on mobile devices for cloud computing [C]//*Proceedings of the* 2009 *ACM Workshop on Cloud Computing Security*. New York, USA, 2009: 127 – 134.

[5] Park Ki-Woong, Han Jaesun, Chung Jae Woong, et al. THEMIS: towards mutually verifiable billing transactions in the cloud computing environment [C]//2010 *IEEE 3rd International Conference on Cloud Computing*. Miami, USA, 2010: 139 – 147.

[6] Hsueh Sue-Chen, Lin Jing-Yan, Lin Ming-Yen. Secure cloud storage for convenient data archive of smart phones [C]//2011 *IEEE 15th International Symposium on Consumer Electronics*. Singapore, 2011: 251 – 258.

[7] Ruiz-Alvarez A, Humphrey M. An automated approach to cloud storage service selection [C]//*Proceedings of the 2nd International Workshop on Scientific Cloud Computing*. New York, USA, 2011: 39 – 48.

[8] Feng Jun, Chen Yu, Ku Wei-Shinn, et al. Analysis of integrity vulnerabilities and a non-repudiation protocol for cloud data storage platforms [C]//2010 39*th International Conference on Parallel Processing Workshops*. San Diego, USA, 2010: 251 – 258.

[9] Ren Wei, Yu Linchen, Gao Ren, et al. Lightweight and compromise resilient storage outsourcing with distributed secure accessibility in mobile cloud computing [J]. *Tsingha Science and Technology*, 2011, **16**(5): 520 – 528.

移动云计算中弹性存储外包方案的安全性分析和改进

刘 晓 蒋 睿

(东南大学信息科学与工程学院,南京 210096)

摘要:分析了移动云计算中弹性存储外包方案,指出该方案中基于共享方案(ShS)存在机密性和完整性缺陷,基于编码方案(CoS)和基于加密方案(EnS)存在完整性缺陷,同时给出针对缺陷的攻击方法. 由此提出了改进安全协议安全共享协议(SShP)、安全编码协议(SCoP)和安全加密协议(SEnP),以克服原协议中存在的安全性缺陷. 采用公钥加密和数字签名,对协议中核心数据加以保护. 安全性分析表明:改进协议可确保用户数据的机密性和完整性,且在保持原方案架构的基础上具有更高安全性. 仿真结果显示改进安全协议的通信开销和原协议相比没有明显增加.

关键词:移动云计算;云存储;安全协议

中图分类号:TP393