

## OOAP: A Novel Authorization Protocol for Access to Sensitive Data in Trusted Cloud Computing Platforms

Dongjun Luo, Xiaoyun Wu, Xinwei Zheng and Yuping Hu

*School of Information Science, Guangdong University of Finance & Economics,  
Guangzhou, 510320, China*

*dongjun.luo@gdufe.edu.cn, agl1975@163.com, caiweihust@163.com and  
okhyp@gdufe.edu.cn*

### **Abstract**

*Cloud computing platforms are usually constructed as trusted virtual platforms based on trusted computing technologies. This is one of the most effective approaches to resolve cloud computing security problems. However, the protection of sensitive data in trusted virtual platforms is an important problem needed to be resolved. In this paper, we proposed a novel authorization protocol. The protocol assembled the functions of OIAP, OSAP and AACP protocols, and prevented all known attacks in existing authorization protocols. Moreover, it satisfied the property of exclusivity and was compatible with TCG TPM command formats. The protocol effectively protects the sensitive data from unauthorized accesses in trusted cloud computing platforms.*

**Keywords:** *authorization protocol, cloud computing, trusted computing, virtual platform, security*

### **1. Introduction**

Cloud computing [1] is a new computing service pattern. It is rapidly developing in industries for its advantages such as conveniences, economies, high extensibilities, and etc. However, it also means customers get out of control of their computations and data once being outsourced to cloud providers. For resolving cloud computing security problems, it is one of the most effective approaches to construct trusted cloud computing virtual platforms based on trusted computing technologies [2]. This can fundamentally protect customers' data and applications in clouds. In virtual platforms, customers' sensitive data are saved in a privileged domain in form of a hierarchy protection system rooting a SRK (Storage Root Key) in TPM or vTPM [3]. When a user process requests sensitive data, the requester must be authorized through a secure channel between the requester and the TPM (or vTPM).

As current typical authorization schemes, TCG (Trusted Computing Group) proposed a series of authorization protocols as follow [4]: a) OIAP (Object-Independent Authorization Protocol) and OSAP (Object-Specific Authorization Protocol) for transmissions of an authorization evidence from requesters to TPMs; b) ADIP (AuthData Insertion Protocol), ADCP (Authdata Change Protocol) and AACP (Asymmetric Authorization Change Protocol) for generations and modifications of authorization data. Based on identity authentications with shared secrets and MACs, the protocols were set up through adding a particular random number called as Rolling Nonce. Among them, the AACP protocol was used to exclusively hold usage rights of an object. It adopted a secret key only known by the owner of an object to protect the

object's new authdata. However, in the ADIP and ADCP protocols, the new authdata of an object were encrypted by the shared secret generated in the OSAP session of the object's father key. This means that the owner of the father key can know the new authdata of the object, and the owner of the TPM can know all authdata in the hierarchy protection system of objects.

The above protocols can defense replay attacks and man-in-the-middle attacks in Dolev-Yao Model by adopting Rolling Nonce and HMAC, and can achieve the exclusiveness of objects through running the AACCP protocol. However, researchers have discovered some attacks in the two basic OIAP and OSAP protocols, such as off-line dictionary attacks due to vulnerable authorization passwords, DoS (Denial of Service) attacks due to lack of authentications for establishing sessions, substitution attacks due to lack of links between shared secrets and input parameters of commands, and packet forgery attacks due to lack of synchronization of sessions [5-7]. Moreover, attackers can control the entire virtual trusted platform, where all domains on the same physical machine share the same SRK and its authdata in the TPM [8]. The attackers knowing the authdata of SRK can forge all child keys generated on the basis of SRK, and forge responses of TPM. And so he can completely control the data storages in the TPM, which seriously threatens the confidentiality and integrity of sensitive data. Researchers proposed a lot of authorization protocols for improvements [5-8]. Unfortunately, these protocols are unable to resolve all the above security problems. In this paper, we propose a novel authorization protocol. It assembles the functions of OIAP, OSAP and AACCP protocols, and effectively prevents the known leaks in these existing authorization protocols. Moreover, it is compatible with the TCG TPM command formats, and needn't the support of symmetric cryptosystem. It has fine properties of security and universality. We call it OOAP (OIAP-OSAP-AACP) protocol.

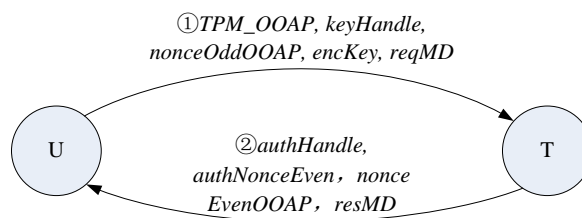
This paper is organized as follows. In Section 2 we describe the OOAP protocol in detail. In Section 3 we analyze the security of the protocol, followed by the solutions to synchronization of sessions in Section 4. The conclusions are given in Section 5.

## 2. Protocol Description

The OOAP protocol is composed of two sections such that a) session establishment and b) command call, as shown in Figure 1 and Figure 2. To present our protocol, let U denote a command requester and let T denote a physical TPM or vTPM. The details of the protocol are listed as follows.

### 2.1. Session Establishment

The process of authorization session establishment of the OOAP protocol is shown in Figure 1.



**Figure 1. The Process of Session Establishment of the OOAP Protocol**

①U → T: *TPM\_OOAP, keyHandle, nonceOddOOAP, encKey, reqMD*

U wants to request T to establish an OOAP session with the *TPM\_OOAP* command. And so, U inputs an object handle *keyHandle* and its authdata *key.usageAuth*, generates a random number *nonceOddOOAP* and a session key *sessionKey* with high entropy, and computes the encrypted key *encKey=Encrypt(pubKey, sessionKey)* and the HMAC value of its request message *reqMD=HMAC(sessionKey, key.usageAuth || keyHandle || nonceOddOOAP || encKey)*. Among them, *pubKey* is a public key of T that U may obtain and only T can hold the corresponding private key *privKey*.

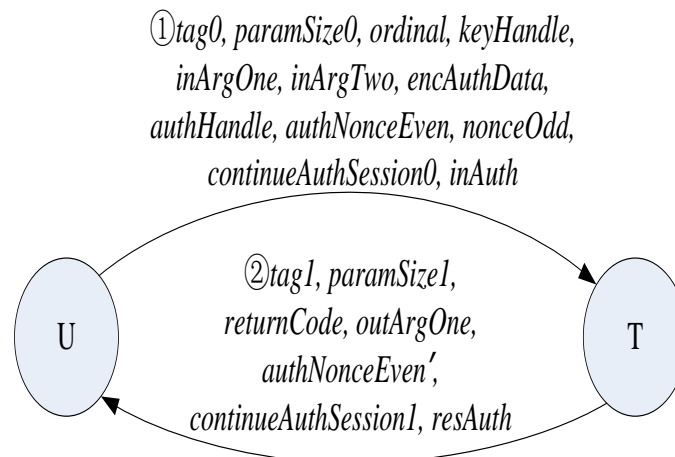
②T → U: *authHandle, authNonceEven, nonceEvenOOAP, resMD*

After receiving Message ①, T decrypts *encKey* and obtain *sessionKey = Decrypt(privKey, encKey)*. Then T reads *key.usageAuth* through *keyHandle* and computes *reqMD'=HMAC(sessionKey, key.usageAuth || keyHandle || nonceOddOOAP || encKey)*. If *reqMD'* equals with *reqMD*, T deems U is authorized. And so T establishes a session storage area denoted with a handle *authHandle*, generates two random numbers *nonceEvenOOAP* and *authNonceEven*, produces a shared secret *sharedSecret=HMAC(sessionKey, nonceOddOOAP || nonceEvenOOAP)*, and computes *resMD=HMAC(sharedSecret, authHandle || authNonceEven || nonceEvenOOAP)*. Then T saves *sharedSecret, authNonceEven, keyHandle* into the storage area denoted by the *authHandle*, and returns Message ② to U.

After receiving Message ②, U produces a shared secret *sharedSecret = HMAC(sessionKey, nonceOddOOAP || nonceEvenOOAP)*, and according to the same approach as T, computes *resMD'=HMAC(sharedSecret, authHandle || authNonceEven || nonceEvenOOAP)*. If *resMD'* equals with *resMD*, U deems T isn't imitative. And so U saves *sharedSecret, authNonceEven, keyHandle* into its session storage area, and denotes the area with the *authHandle* in Message ②.

## 2.2. Command Call

The process of command call of the OOAP protocol is shown in Figure 2.



**Figure 2. The Process of Command Call of the OOAP Protocol**

①U → T: *tag0, paramSize0, ordinal, keyHandle, inArgOne, inArgTwo, encAuthData, authHandle, authNonceEven, nonceOdd, continueAuthSession0, inAuth*

U wants to request T to run the *ordinal=TPM\_Example* command through a session handle *authHandle*, where U will visit an inner resource pointed by an object handle *keyHandle* in TPM and (optionally) set up an authdata for a new entity. For this purpose, U generates an authdata *newEntity.usageAuth* of the new entity, and computes  $encAuthData = XOR(newEntity.usageAuth, SHA1(sharedSecret \parallel authNonceEven))$ . Then U generates a random number *nonceOdd*, and check whether there exists the *keyHandle* in its session storage area denoted by the *authHandle*. If exists, U computes  $inAuth = HMAC(sharedSecret, inParamDigest \parallel inAuthSetupParams)$ ; or else, U inputs the *keyHandle* and the object's authdata *key.usageAuth*, and computes  $inAuth = HMAC(sharedSecret, key.usageAuth \parallel inParamDigest \parallel inAuthSetupParams)$ , where  $inParamDigest = SHA1(ordinal \parallel inArgOne \parallel inArgTwo \parallel encAuthData)$  and  $inAuthSetupParams = (authHandle \parallel authNonceEven \parallel nonceOdd \parallel continueAuthSession0)$ . Then, U sends the *TPM\_Example* command to T according to the TCG serialization format of parameters.

②T → U: *tag1, paramSize1, returnCode, outArgOne, authNonceEven', continueAuthSession1, resAuth*

After receiving Message ①, T checks whether there exists the *keyHandle* in its session storage area denoted by the *authHandle*. If exists, T reads the *sharedSecret* and *authNonceEven* from its session storage area, and according to the same approach as U, computes  $inAuth' = HMAC(sharedSecret, inParamDigest \parallel inAuthSetupParams)$ ; or else, T reads the *authNonceEven* and *key.usageAuth* respectively through the *authHandle* and *keyHandle*, and computes  $inAuth' = HMAC(sharedSecret, key.usageAuth \parallel inParamDigest \parallel inAuthSetupParams)$  according to the same approach as U. If *inAuth'* equals with *inAuth*, T deems U is valid, and saves the *keyHandle* into its session storage area if the object handle hasn't been saved. Then T decrypts the *encAuthData*, that is,  $newEntity.usageAuth = XOR(encAuthData, SHA1(sharedSecret \parallel authNonceEven))$ ; runs the *TPM\_Example* command and updates its *authNonceEven* into the *authNonceEven' = authNonceEven + 1*; computes  $resAuth = HMAC(sharedSecret, outParamDigest \parallel outAuthSetupParams)$ , where  $outParamDigest = SHA1(returnCode \parallel ordinal \parallel outArgOne)$  and  $outAuthSetupParams = (authHandle \parallel authNonceEven' \parallel nonceOdd \parallel continueAuthSession1)$ ; and generates Message ② to response U. If the *continueAuthSession0 = true*, T continues the session.

After receiving Message ②, U computes  $resAuth' = HMAC(sharedSecret, outParamDigest \parallel outAuthSetupParams)$  according to the same approach as T. If *resAuth'* equals with *resAuth*, U accepts the above response, saves the *keyHandle* if the object handle hasn't been saved into its session storage area, and also updates its *authNonceEven* into the *authNonceEven'*.

### 3. Protocol Analysis

The OOAP protocol satisfies the following important properties.

1) The OOAP protocol has functions of both OSAP and OIAP protocols. In the proposed protocol, due to the mechanism of session management, a session can be used to authorize to run lots of commands on the same object without inputting authdata many times, also to authorize to visit lots of objects with compulsively inputting authdata only once for each different object. Resorting to the mechanism of session management, the proposed protocol can guard against substitution attacks taking place in the OSAP protocol.

2) The OOAP protocol satisfies the property of exclusivity. In the proposed protocol, the shared secret *sharedSecret* used to encrypt the authdata of a new entity is generated based on the session key *sessionKey* that U provides, which avoids the inferences of the owner of the entity's father key. This makes the new entity be exclusive. Due to the property, the proposed protocol can solve the security problems caused by sharing the SRK and its authdata in TPM in virtual trusted platforms.

3) The OOAP protocol is capable to effectively prevent replay attacks, off-line dictionary attacks and DoS (Denial of Service) attacks. In the proposed protocol, T updates the *authNonceEven* by adding 1 for sharing, which decreases the collision probability of random numbers. This effectively avoids replay attacks to T. When U is demanded to input authdata of a visited entity, the authdata will be converted into a HMAC value by a session key *sessionKey* with high entropy or a shared secret *sharedSecret*. This effectively avoids off-line dictionary attacks caused by vulnerable authorization passwords (authdata). Also, in the OOAP protocol, only authenticated and authorized users are allowed to set up authorization sessions. This avoids plenty of unnecessary computations that TPM costs to establish sessions once receiving requests in the OSAP and OIAP protocols, and avoids the waste of resources in TPM.

In addition, the proposed protocol is also capable to prevent the imitation of T. Moreover, the proposed protocol has fine compatibility due to none supports of symmetric cryptosystem and be compatible with TCG TPM command formats.

#### 4. Discussion of Synchronization Processing

In the OOAP protocol, T defends itself against replay attacks by adopting *authNonceEven* + 1 at every turn. However, the delay utilization of the *authNonceEven* value easily makes both communication sides save inconsistent values of the *authNonceEven*, which leads T to fail to verify the *inAuth*. And so the OOAP protocol especially puts the parameter *authNonceEven* into the Message ① in Figure 2 to synchronize the *authNonceEven* values. There exist 2 possible causes which result in the verification failure: a) attackers replay the Message ① in Figure 2; b) U sends the Message ① in Figure 2 again with not updated *authNonceEven* values because it receives a forged or real package of network errors for responses. For case a), T doesn't need do anything. And for case b), T must send an alert to synchronize the *authNonceEven* values with U. However, it isn't simple to distinguish the 2 cases. If T doesn't deal with case b), U will deem the new entity and its authdata to be unsuccessfully set up, but inversely in fact. This wastes the storage and computing resources in TPM. And so, in the proposed protocol, T always sends an alert for the *authNonceEven* synchronization to U once the *authNonceEven* values of both sides are inconsistent, regardless of the 2 cases. This method simplifies the complexity of the OOAP protocol and effectively prevents packet forgery attacks.

#### 5. Conclusions

The OOAP protocol assembles the functions of OIAP, OSAP and AACP, and prevents all the known attacks in existing authorization protocols such as the OIAP, OSAP and etc. It is capable to defense off-line dictionary attacks, DoS attacks, substitution attacks and packet forgery attacks. Moreover, it satisfies the property of exclusivity, and is compatible with the TCG TPM command formats. It has fine properties of security and universality, which effectively protects the sensitive data from unauthorized accesses in trusted cloud computing platforms.

## Acknowledgements

This work was supported by Humanities and Social Science Planning Fund of Ministry of Education of China (No.13YJAZH099), Guangdong University of Finance & Economics Foundation (No.11ZD52001).

## References

- [1] I. Foster, Y. Zhao, T. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared", Proceeding of Grid Computing Environments Workshop, (2008) November 12-16, Austin, TX, United states.
- [2] N. Santos, K. P. Gummadi and R. Rodrigues, "Towards trusted cloud computing", Proceeding of the 2009 Workshop on Hot Topics in Cloud Computing, (2009) June 14-19, San Diego, CA, United states.
- [3] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer and L. van Doorn, "vTPM: Virtualizing the trusted platform module", Proceeding of the 15th USENIX Security Symposium, (2006) July 31-August 4, Vancouver, B.C., Canada.
- [4] "Trusted Computing Group, TCG specification architecture overview revision 1.4", (2007), <https://www.trustedcomputinggroup.org/groups/TCG.1.4.Architecture.Overview.pdf>.
- [5] X. Zhang, X. F. Zhang and C. X. Shen, "A new authorization protocol for trusted computing", Proceeding of 1st International Symposium on Data, Privacy and E-Commerce, (2007) November 1-3, Chengdu, China.
- [6] W. Liu, M. Tan, and X. S. Chen, "Security analysis and improvement of the TPM's two main authorization protocols", Journal of Computer Science, vol. 3, no. 35, (2008).
- [7] Y. P. Chi, Z. P. Li, Z. Z. Wei and Y. Fang, "Analysis and improvement of trusted computing authorization protocol", Journal of Harbin Institute of Technology, vol. 3, no. 44, (2012).
- [8] M. Liang and C. W. Chang, "Analysis and improvement of TPM object access authorization protocol in virtualization environment", Journal of Chinese Computer Systems, vol. 7, no. 33, (2012).

## Authors



**Dongjun Luo**, he received his M.S. degree in Computer Software and Theory from Sun Yat-Sen University, Guangzhou, China, in 2004 and Ph.D. degree in Information Security from South China University of Technology, Guangzhou, China, in 2014. He is a lecturer with School of Information Science, Guangdong University of Finance & Economics. His main research interests include trusted computing, cloud security and electronic commerce security.



**Xiaoyun Wu**, he received his Ph.D. degrees from Sun Yat-Sen University, Guangzhou, China, in 2006. He is currently an associate professor with School of Information Science, Guangdong University of Finance & Economics. His current research interests include image processing and information security.



**Xinwei Zheng**, she respectively received her Master and Ph.D. degree in Computer Science from Huazhong University of Science and Technology, Wuhan, China, in 2003 and 2008. She is a lecturer with School of Information Science, Guangdong University of Finance & Economics. Her current research interests include Multipath Routing, computer network technology and stream media technology.



**YuPing Hu**, he received his Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2005. He is a professor with School of Information Science, Guangdong University of Finance & Economics, since 2006. His current research interests include digital watermarking, image processing, multimedia and network security.

