SCIENCE CHINA
Information Sciences

• HIGHLIGHT •

March 2018, Vol. 61 038106:1–038106:3
doi: 10.1007/s11432-017-9124-0

# Secure searchable public key encryption against insider keyword guessing attacks from indistinguishability obfuscation

Lixue SUN[1], Chunxiang XU[1*], Mingwu ZHANG[2], Kefei CHEN[3,4] & Hongwei LI[1]

[1]*Center for Cyber Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu* 611731*, China;*
[2]*School of Computer Science, Hubei University of Technology, Wuhan* 430068*, China;*
[3]*School of Science, Hangzhou Normal University, Hangzhou* 310036*, China;*
[4]*Laboratory of Science and Technology on Communication Security, Chengdu* 610041*, China*

Received 23 May 2017/Accepted 2 June 2017/Published online 23 August 2017

**Citation** 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(3): 038106, doi: 10.1007/s11432-017-9124-0

Public key encryption with keyword search (PEKS) enables one to search encrypted data containing some keyword without decrypting the data in public key setting. Boneh et al. [1] introduced the notion of PEKS, but later Byun et al. [2] showed the scheme in [1] is vulnerable to the outsider/insider keyword guessing attacks (KGA). Many previous PEKS studies have tackled the problem of keyword privacy against outsider KGA, while how to ensure keyword privacy against insider KGA launched by an untrusted cloud sever receives little attention. Moreover, the majority of PEKS schemes assume that the cloud server is honest-but-curious. In other researches, the server will honestly execute the search protocol and only attempt to infer sensitive information about the users. However, it is not always the case in the real world. The cloud server may be malicious, which may not honestly perform the search operation.

The primary reason leading to insider KGA is the public generation of the searchable ciphertext. For a received trapdoor, the untrusted cloud server can guess a keyword, generate the corresponding searchable ciphertext, and then test whether the guessing keyword is the one underlying the trapdoor. Recently, Jiang et al. [3] proposed a PEKS scheme based on identity based cryptosystem, which achieves keyword privacy against insider KGA by imposing a trusted third party who generates and distributes the secret keys for the senders. In this article, we explore whether a PEKS scheme against insider KGA can be built based on different public key cryptosystems, such as PKI-based, identity-based, or certificateless cryptosystem. Two types of inside attackers are considered when devising our scheme: an honest-but-curious server or malicious server.

We adopt signcryption algorithm (or signature-then-encryption, or encryption-then-signature) when generating the searchable ciphertext. Therefore, without the sender's secret key, the cloud server cannot generate a legitimate searchable ciphertext to match a received trapdoor. Our crucial challenge is how to generate the matched trapdoor on the receiver side. Inspired by a cryptographic primitive, indistinguishability obfuscator ($i\mathcal{O}$) [4–7], we define the trapdoor as a ci-

* Corresponding author (email: chxxu@uestc.edu.cn)
The authors declare that they have no conflict of interest.

phertext of the interested keyword, the sender's public key, and a random number, encrypted using a symmetric encryption algorithm (e.g., AES). Thus, the searchable ciphertext and the trapdoor can be matched by executing decryption and unsigncryption operations in an obfuscated program embedded in the receiver's symmetric key and decryption key. Inversely, without using $i\mathcal{O}$, it will be very difficult to match the searchable ciphertext generated using a signcryption algorithm and the trapdoor generated using a symmetric encryption algorithm. Additionally, if the server is malicious, a message authentication code (MAC) function is used in the obfuscated program to enable the authenticity check of the returned search results. Therefore, we successfully design a PEKS scheme against insider KGA from $i\mathcal{O}$ based on any public key encryption system.

*Notations.* Let $\mathrm{SE} = (\mathrm{SE.KG}(\lambda), \mathrm{SE.Enc}(K, m), \mathrm{SE.Dec}(K, m))$ be an IND-CPA (indistinguishability against chosen plaintext attacks) secure symmetric encryption scheme, $\mathrm{SC} = (\mathrm{SC.KG}(\lambda)$, $\mathrm{SC.Signcrypt}(\mathrm{SK}, \mathrm{PK}', m)$, $\mathrm{SC.Unsigncrypt}(\mathrm{VK}, \mathrm{SK}', C))$ be a PKI-based signcryption scheme satisfying confidentiality (indistinguishability against adaptive chosen ciphertext attacks (IND-CCA2)) and unforgeability (existential unforgeability against adaptive chosen message attacks (EUF-CMA)), and $i\mathcal{O}$ be a secure indistinguishability obfuscator. Here, $\lambda$, $m$, $C$, $K$, SK, PK$'$, VK and SK$'$ denote respectively the security parameter, message, signcryption ciphertext, symmetric key, the sender's secret key, the receiver's public key, the sender's public key and the receiver's secret key.

**Remark 1.** The signcryption scheme used in our PEKS scheme can also be an identity-based one or a certificateless one, so that our PEKS scheme against insider KGA can be built based on different public key cryptosystems (e.g., PKI-based, identity-based, or certificateless cryptosystem).

*The construction.* We give our scheme in terms of two types of inside attackers: an honest-but-curious server or malicious server. If the server is malicious, it requires extra a verification algorithm to verify if the returned search results are valid.

**Setup**($\lambda$): Taking as input the security parameter $\lambda$, it outputs the system parameters params.

**KeyGen**(params): Taking as input $\lambda$ and params, it runs SC.KG and SE.KG, and outputs a public/secret key pair $(\mathrm{PK_S}, \mathrm{SK_S})$ for the sender, a public/secret key pair $(\mathrm{PK_R}, \mathrm{SK_R})$ and a symmetric key $K$ for the receiver if the server is honest-but-curious. It needs to generate extra a MAC key $K_1$ for the receiver, if the server is malicious.

**Encrypt**$(\mathrm{params}, \mathrm{PK_R}, \mathrm{SK_S}, w)$: Taking as input params, $\mathrm{PK_R}$, $\mathrm{SK_S}$, and $w$, it outputs the searchable ciphertext

$$\mathrm{PEKS} = \mathrm{SC.Signcrypt}(\mathrm{PK_R}, \mathrm{SK_S}, w).$$

Finally, the sender uploads PEKS to the cloud server.

**Trapdoor**$(\mathrm{params}, K, \mathrm{PK_S'}, N, w')$: This algorithm takes as input params, $K$, $\mathrm{PK_S'}$, $w'$ and a random number $N$, and outputs the trapdoor for $\mathrm{PK_S'}$ on $w'$ as

$$T_{w'} = \mathrm{SE.Enc}(K, w' \| \mathrm{PK_S'} \| N).$$

Then the receiver sends the trapdoor $T_{w'}$ to the cloud server.

**Remark 2.** Before the trapdoor generation algorithm, the receiver needs to pre-construct a test circuit embedded in $\mathrm{SK_R}$ and $K$. If the server is malicious, the MAC key $K_1$ is also embedded in the test circuit. The procedure of the test circuit for a malicious sever is demonstrated in Figure 1. Then the receiver obfuscates it to get $i\mathcal{O}(\mathcal{P}_t)$. Finally, the receiver sends the obfuscated circuit to the cloud server. If the server is honest-but-curious, the 3rd step in Figure 1 is replaced by "If $w' = w''$, output 1; Else output 0".

**Test**$(\mathrm{params}, T_{w'}, \mathrm{PEKS})$: The cloud server puts params, PEKS and $T_{w'}$ into the obfuscated circuit and runs it. If the server is honest-but-curious, this algorithm outputs 1 if and only if $\mathrm{PK_S} = \mathrm{PK_S'}$ and $w' = w''$; Otherwise, outputs 0. If the server is malicious, this algorithm outputs (PEKS, Mac) if and only if $\mathrm{PK_S} = \mathrm{PK_S'}$ and $w' = w''$; Otherwise, outputs 0.

**Verify**$(\mathrm{PEKS}, T_{w'}, \mathrm{Mac}, K_1)$: If the server is malicious, the receiver verifies the hash value by $\mathrm{MAC}(K_1, \mathrm{PEKS} \| T_{w'}) \overset{?}{=} \mathrm{Mac}$. If the verification is passed, it outputs 1; Otherwise, it outputs 0.

*Security.* We prove our construction above satisfies semantic security against chosen keyword attacks (SS-CKA) and keyword guessing attacks (SS-KGA), and verifiability of the returned search results. The security of SS-CKA and SS-KGA captures respectively that the searchable ciphertext and the trapdoor do not reveal any information about the underlying keyword to the adversary.

```
Test circuit $\mathcal{P}_t$:
Input: params, PEKS, $T_{w'}$.
Constants: $\mathrm{SK_R}$, $K$, $K_1$.
1. Compute $w' \| \mathrm{PK_S'} \| N = \mathrm{SE.Dec}(K, T_{w'})$.
2. Compute $w'' = \mathrm{SC.Unsigncrypt}(\mathrm{PK_S'}, \mathrm{SK_R}, \mathrm{PEKS})$.
If algorithm SC.Unsigncrypt fails, output 0.
3. If $w' = w''$, compute $\mathrm{Mac} = \mathrm{MAC}(K_1, \mathrm{PEKS} \| T_{w'})$,
and output (PEKS, Mac); Else output 0.
```

**Figure 1** Test circuit $\mathcal{P}_t$ for a malicious server.

**Theorem 1.** Assuming SE is an IND-CPA secure symmetric encryption scheme, SC is a PKI-based signcryption scheme satisfying confidentiality (IND-CCA2) and unforgeability (EUF-CMA), and $i\mathcal{O}$ is a secure indistinguishability obfuscator, then our PEKS scheme is semantically secure against CKA in the random oracle model.

*Proof.* Refer to Appendix A.

**Theorem 2.** Assuming SE is an IND-CPA secure symmetric encryption scheme, SC is a PKI-based signcryption scheme satisfying confidentiality (IND-CCA2) and unforgeability (EUF-CMA), and $i\mathcal{O}$ is a secure indistinguishability obfuscator, then our PEKS scheme is semantically secure against KGA in random oracle model.

*Proof.* Refer to Appendix B.

**Theorem 3.** Assuming MAC($\cdot$) is a secure MAC, and $i\mathcal{O}$ is a secure indistinguishability obfuscator, then our PEKS scheme satisfies verifiability if the cloud server is malicious.

*Proof.* Refer to Appendix C.

*Performance.* We first analyze the security of our scheme in terms of CKA, outsider/insider KGA, and any public key cryptosystem (PKCS). Our scheme against insider KGA can be built based on PKI-based cryptosystem, identity-based cryptosystem, or certificateless cryptosystem.

Then, we give the cost analyses on trapdoor generation which affects the receiver's experience when searching over the encrypted data. The trapdoor generation algorithm in our scheme includes only a symmetric encryption, which achieves a high efficiency on the receiver side. The experiment results are given in Appendix D.

*Conclusion.* A traditional PEKS scheme suffers from insider KGA launched by an untrusted server. In this article, we develop a novel approach to resist insider KGA in PEKS. Our scheme has comparable advantage when compared to other related works and is proved to be semantically secure in the random oracle model.

**Acknowledgements** This work was supported in part by National Natural Science Foundation of China (Grant No. 61370203) and National Science and Technology Major Project (Grant No. 2017YFB0802003).

**Supporting information** Appendixes A–D. 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 Boneh D, Di Crescenzo G, Ostrovsky R, et al. Public key encryption with keyword search. In: Advances in Cryptology-Eurocrypt 2004. Berlin: Springer, 2004. 506–522
2 Byun J W, Rhee H S, Park H A, et al. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Proceedings of Workshop on Secure Data Management, Seoul, 2006. 75–83
3 Jiang P, Mu Y, Guo F C, et al. Online/offline ciphertext retrieval on resource constrained devices. Comput J, 2016, 59: 955–969
4 Cheng R, Yan J B, Guan C W, et al. Verifiable searchable symmetric encryption from indistinguishability obfuscation. In: Proceedings of ACM Symposium on Information, Computer and Communications Security, Singapore, 2015. 621–626
5 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 Foren Secur, 2016, 12: 676–688
6 Barak B, Goldreich O, Impagliazzo R, et al. On the (im) possibility of obfuscating programs. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology, Santa Barbara, 2001. 1–18
7 Garg S, Gentry C, Halevi S, et al. Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS), Berkeley, 2013. 40–49