Information Processing Letters 115 (2015) 957–961

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

On security against the server in designated tester public key encryption with keyword search

Zhi-Yi Shao [a,b,*], Bo Yang [b]

[a] The Library, Shaanxi Normal University, Xi'an 710062, China
[b] School of Computer Science, Shaanxi Normal University, Xi'an 710062, China

ARTICLE INFO

Article history:
Received 31 March 2014
Received in revised form 16 June 2015
Accepted 16 July 2015
Available online 22 July 2015
Communicated by S.M. Yiu

Keywords:
Searchable encryption
Keyword guessing attacks
Cryptography

ABSTRACT

The offline keyword guessing attack (KG attack) is a new security threat to the designated tester public key encryption with keyword search (dPEKS). Many techniques have been proposed to resist such an attack. However, all the schemes which are secure against KG attacks have not solved the problem that the KG attacker is the server. We redefine the security of dPEKS against KG attacks and propose *IND-KGA-SERVER* security. Then based on the existence of the Certificate Authority of the Public Key Infrastructure and the deterministic digital signature, we demonstrate how to construct secure dPEKS when the KG attacker is the server. Our solution is a bootstrap from *IND-KGA* secure dPEKS to the one of *IND-KGA-SERVER* security.

1. Introduction

The notion of public key encryption with keyword search (PEKS) was put forward by Boneh et al. [1]. In such a scheme, the email sender sends an email to the email receiver through the email server. The body of this email is encrypted using an ordinary encryption scheme. But the keywords are encrypted using a searchable encryption scheme by which the server could search for a specific keyword using a trapdoor without learning its corresponding plaintext.

In PEKS, although an attacker obtains the trapdoor of any keyword, he is still unable to distinguish the PEKS ciphertext of one keyword from that of another. This is *IND-CKA* security proposed by Boneh et al. in a sense of semantic security.

However, *PEKS* needs a secure channel between the server and the receiver. To remove the secure channel, Baek et al. proposed dPEKS (designated tester PEKS) [2]. In dPEKS, only the designated server can test which dPEKS ciphertext is related to the given trapdoor by using his secret key. But such a security cannot necessarily prevent the attacker from learning the keyword in other ways.

Recently, Byun et al. showed that an attacker could obtain the keyword in dPEKS by launching "off-line keyword guessing attacks" (KG attacks) [3]. The reason is that keywords are often chosen from a small space and of low entropy. The attacker thus could test with all the keywords one by one. If some keyword and the obtained trapdoor satisfy a certain equation, the attacker guesses the keyword. Byun et al. showed the attacker succeeds with non-negligible probability, which encourages many researchers to solve such a problem.

Rhee et al. introduced the "trapdoor indistinguishability", and showed that this is a sufficient condition against KG attacks [4,5].

However, their scheme [5] runs in the random oracle model, and the attacker cannot propose the Test query in their security model. So Fang et al. improved the security model, and proposed in the standard model a dPEKS

* Corresponding author.
E-mail addresses: shaozy@snnu.edu.cn (Z.-Y. Shao), byang@snnu.edu.cn (B. Yang).

http://dx.doi.org/10.1016/j.ipl.2015.07.006
0020-0190/

scheme which is secure against KG attacks. This scheme is called *IND-KGA* secure.

Fang et al.'s scheme satisfies strong security. However, neither their work nor the former ones solve the problem that the KG attacker is the server.

In this letter, we strengthen the security model against KG attacks and solve the security problem against KG attacks when the attacker is the server.

2. Preliminaries

2.1. Certificate authentication (CA)

CA is the most important component of PKI (Public Key Infrastructure). Its task is to generate, publish, revoke, and archive the digital certificates [6]. CA has been used in design of many security protocols such as secure socket layer, secure electronic transaction, and identity authentication. The reason is that CA has the ability to demonstrate the identity of the entities involved, and the relationship between their key pairs and their identity information [7]. This is why we use CA.

2.2. Deterministic signature

The RSA signature is deterministic and runs as follows,

(1) On input a security parameter λ, *Gen* outputs the public key $pk = (N, e)$ and the secret key $sk = (N, d)$.
(2) On input the secret key sk and the message $m \in \mathbb{Z}_N^*$, *Sign* outputs the signature $\sigma = m^d \bmod N$.
(3) On input pk, and σ, *Veri* checks $m \stackrel{?}{=} \sigma^e \bmod N$. If this equation holds, output 1, and 0 otherwise.

We should first caution the reader that RSA signature is not secure because of the following reasons [8].

(1) Anyone who obtains the corresponding public key alone can forge signatures. Given pk, the attacker chooses arbitrarily $\sigma \in \mathbb{Z}_N^*$ and computes $m = \sigma^e \bmod N$. The attacker obtains a forgery (m, σ).
(2) The attacker can easily forge a signature on an arbitrary message m. The attacker randomly chooses $m_1 \in \mathbb{Z}_N^*$ and sets $m_2 = (m/m_1) \bmod N$. He obtains signatures σ_1 and σ_2 on m_1 and m_2 respectively. Then the attacker computes $\sigma = (\sigma_1 \cdot \sigma_2) \bmod N$ and obtains the forgery (m, σ).

The attacker forges a valid signature on a message m as above. However, this does not prevent us from using it securely with the CA, as the reason we use them is to distinguish the server from others.

The property of RSA signature of which we will take advantage is that using the same secret key to sign on a message twice, we obtain two equal signatures. That is, RSA signature is deterministic.

3. Our security model

We define the security model against KG attacks by an experiment (game) played between the attacker $\mathcal{A}$ and an imagined challenger $\mathcal{B}$ [8]. $\mathcal{A}$ has the abilities to propose Trapdoor queries, to generate dPEKS ciphertext on his own, and to obtain the server's secret key. Trapdoor queries describe that $\mathcal{A}$ can obtain any trapdoors sent in an unsecure channel. Anyone who obtains the corresponding public keys could compute the dPEKS ciphertext on his own, and $\mathcal{A}$ could too. We also provide $\mathcal{A}$ with the server's secret key. This implies $\mathcal{A}$ could be the server. But at the same time, allowing an outside attacker to access to the server's secret key will make the attacker more powerful. In such an experiment, $\mathcal{B}$ establishes the algorithm, acts as the receiver, it interacts with $\mathcal{A}$ and tries to see whether $\mathcal{A}$ could succeed in guessing the keyword. We call such a security experiment *IND-KGA-SERVER* experiment and denote it by $Exp_{\mathcal{A},\Pi}^{IND\text{-}KGA\text{-}SERVER}(\lambda)$ where λ is the security parameter and Π is the dPEKS scheme under consideration. The experiment runs as follows.

IND-KGA-SERVER Experiment $Exp_{\mathcal{A},\Pi}^{IND\text{-}KGA\text{-}SERVER}(\lambda)$**.**

(1) **Setup.** *$\mathcal{B}$ runs $GlobalSetup(\lambda)$ to generate the global parameter $\mathcal{GP}$, runs $KeyGen_{server}(\mathcal{GP})$ to obtain the server's key pair (pk_s, sk_s), and runs $KeyGen_{receiver}(\mathcal{GP})$ to obtain the receiver's key pair (pk_r, sk_r). $\mathcal{B}$ sends (pk_s, sk_s) and pk_r to $\mathcal{A}$. The reason of sending sk_s to $\mathcal{A}$ is to capture the server's feature. But this also gives an outside attacker more power.*
(2) **Query Phase 1.** *$\mathcal{A}$ makes the following queries.*
Trapdoor Query. $\mathcal{A}$ adaptively asks for the trapdoor T_w of any keyword $w \in \mathcal{KS}$. $\mathcal{B}$ responds with the trapdoor $T_w = Trapdoor(\mathcal{GP}, pk_s, sk_r, w)$.
(3) **Challenge.** *At some point, $\mathcal{A}$ finishes step 2, and outputs the keyword pair (w_0, w_1) which is to be challenged. Neither w_0 nor w_1 was previously queried in step 2. $\mathcal{A}$ sends (w_0, w_1) to $\mathcal{B}$. $\mathcal{B}$ randomly chooses $b \in \{0, 1\}$, generates the corresponding trapdoor T_{w_b}, and sends T_{w_b} to $\mathcal{A}$.*
(4) **Query Phase 2.** *$\mathcal{A}$ continues to query as in step 2. The only restriction is that $\mathcal{A}$ should not query on w_0 and w_1.*
(5) **Guess.** *$\mathcal{A}$ outputs his guess b'. If $b' = b$, $\mathcal{A}$ wins the game and outputs 1, and 0 otherwise.*

We define the advantage of $\mathcal{A}$ in this experiment by $Adv_{\mathcal{A}}^{IND\text{-}KGA\text{-}SERVER}(\lambda) = |Pr[b' = b] - 1/2|$.

Definition 1 *(IND-KGA-SERVER security).* A dPEKS scheme Π is said to be IND-KGA-SERVER secure, if for any probabilistic polynomial-time attacker $\mathcal{A}$ including the server, $Adv_{\mathcal{A}}^{IND\text{-}KGA\text{-}SERVER}(\lambda)$ is negligible.

Lemma 2. *For an IND-KGA secure dPEKS, if it is IND-KGA secure against the server, and IND-KGA secure against the outside attacker who obtains the server's secret key, then such a dPEKS scheme is IND-KGA-SERVER secure.*

Proof. The only difference between *IND-KGA* security and *IND-KGA-SERVER* security lies in the description of the at-