Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

12-2016

Server-aided public key encryption with keyword search

Rongman CHEN

Yi MU

Guomin YANG

Fuchun GUO

Xinyi HUANG

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Information Security Commons

Citation

CHEN, Rongman; MU, Yi; YANG, Guomin; GUO, Fuchun; HUANG, Xinyi; WANG, Xiaofen; and WANG, Yongjun. Server-aided public key encryption with keyword search. (2016). *IEEE Transactions on Information Forensics and Security*. 11, (12), 2833-2842.

Available at: https://ink.library.smu.edu.sg/sis_research/7359

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Author

Rongman CHEN, Yi MU, Guomin YANG, Fuchun GUO, Xinyi HUANG, Xiaofen WANG, and Yongjun WANG

This journal article is available at Institutional Knowledge at Singapore Management University: https://ink.library.smu.edu.sg/sis_research/7359

Server-Aided Public Key Encryption With Keyword Search

Rongmao Chen, Yi Mu, *Senior Member, IEEE*, Guomin Yang, *Member, IEEE*, Fuchun Guo, Xinyi Huang, Xiaofen Wang, and Yongjun Wang

Abstract—**Public key encryption with keyword search (PEKS) is a well-known cryptographic primitive for secure searchable data encryption in cloud storage. Unfortunately, it is inherently subject to the (inside) offline keyword guessing attack (KGA), which is against the data privacy of users. Existing countermeasures for dealing with this security issue mainly suffer from low efficiency and are impractical for real applications. In this paper, we provide a practical and applicable treatment on this security vulnerability by formalizing a new PEKS system named server-aided public key encryption with keyword search (SA-PEKS). In SA-PEKS, to generate the keyword ciphertext/trapdoor, the user needs to query a semitrusted third-party called keyword server (KS) by running an authentication protocol, and hence, security against the offline KGA can be obtained. We then introduce a universal transformation from any PEKS scheme to a secure SA-PEKS scheme using the deterministic blind signature. To illustrate its feasibility, we present the first instantiation of SA-PEKS scheme by utilizing the Full Domain Hash RSA signature and the PEKS scheme proposed by Boneh *et al.* in Eurocrypt 2004. Finally, we describe how to securely implement the client-KS protocol with a rate-limiting mechanism against online KGA and evaluate the performance of our solutions in experiments.**

Index Terms—**Public key encryption with keyword search, server-aided, off-line keyword guessing attack.**

I. Introduction

CLOUD storage outsourcing is of increasing interest in recent years for enterprises and organizations to reduce the burden of maintaining big data. In reality, end users may prefer to encrypt their outsourced data for privacy protection as they may not entirely trust the cloud storage server. This makes deployment of traditional data utilization service, such as plaintext keyword search over textual data or query over database, a difficult task. One of the typical solutions is the searchable encryption which allows the user to search and retrieve the encrypted data, and meanwhile preserve the data privacy. Searchable encryption can be realized in either symmetric [1], [2] or asymmetric encryption setting [3], [4]. The symmetric searchable encryption (SSE) is proposed by Song *et al.* [1] and later a formal treatment by Curtmola *et al.* [2]. Despite the high efficiency in SSE schemes, they suffer from complicated secret key distribution problem. Searchable encryption in public key setting, originating from store-and-forward system, such as email system, in which a receiver can search data encrypted under the receiver's public key on an outsourced storage system, is initiated by Boneh *et al.* [3]. They firstly introduced a more flexible primitive, namely Public Key Encryption with Keyword Search (PEKS) that enables a user to search encrypted data in the asymmetric encryption setting. In a PEKS system, using the receiver's public key, the sender attaches some encrypted keywords (referred to as PEKS ciphertexts) with the encrypted data. The receiver then sends the trapdoor of a to-be-searched keyword to the server for data searching. Given the trapdoor and the PEKS ciphertext, the server can test whether the keyword underlying the PEKS ciphertxt is equal to the one selected by the receiver. If so, the server sends the matching encrypted data to the receiver.

A. Motivation of This Work

Unfortunately, despite being free from secret key distribution, PEKS schemes suffer from an inherent security problem regarding the keyword privacy, namely (inside) off-line Keyword Guessing Attack (KGA). Specifically, given a trapdoor, the adversarial server can choose a guessing keyword from the keyword space and then use the keyword to generate a PEKS ciphertext. The server then can test whether the guessing keyword is the one underlying the trapdoor. This *guessing-then-testing* procedure can be repeated until the correct keyword is found. As the keyword always could leak some sensitive information of the user data, it is therefore of practical importance to overcome this security threat for secure and searchable encrypted data outsourcing.

In [5], Xu *et al.* proposed the notion of Public-key Encryption with Fuzzy Keyword Search (PEFKS) where

Manuscript received November 25, 2015; revised March 10, 2016, May 5, 2016, and July 14, 2016; accepted July 19, 2016. Date of publication August 10, 2016; date of current version October 11, 2016. The work of Y. Wang was supported by the National Natural Science Foundation of China under Grant 61472439. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Giuseppe Persiano. *(Corresponding authors: Rongmao Chen; Xiaofen Wang.)*

R. Chen is with the Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia, and also with the College of Computer, National University of Defense Technology, Changsha 410003, China (e-mail: rc517@uowmail.edu.au).

Y. Mu, G. Yang, and F. Guo are with the Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: ymu@uow.edu.au; gyang@uow.edu.au; fuchun@uow.edu.au).

X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350000, China (e-mail: xyhuang@fjnu.edu.cn).

X. Wang is with School of Computer Science and Engineering and Big Data Research Center, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China (e-mail: wangxuedou@sina.com).

Y. Wang is with the College of Computer, National University of Defense Technology, Changsha 410003, China (e-mail: wwyyjj1971@126.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2599293

1556-6013 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

each keyword corresponds to an exact trapdoor and a fuzzy trapdoor. The server is only provided with the fuzzy trapdoor and thus can no longer learn the exact keyword. However, in their scheme, the malicious server is still able to identify a small set the underlying keyword belongs to and thus the keyword privacy is not well preserved from the server. On the other hand, their scheme is impractical as the receiver has to locally find the matching ciphertext by using the exact trapdoor to filter out the non-matching ones from the set returned from the server. Another work by Chen et al. [6], [7] proposed a new framework of PEKS, namely Dual-Server Public Key Encryption with Keyword Search (DS-PEKS) to achieve the security against inside KGA. Their central idea is to disallow the stand-alone testing of PEKS by splitting the testing functionality of the PEKS system into two parts which are handled by two independent servers. Therefore, the security against the off-line KGA can be obtained as long as the two servers do not collude. Nevertheless, the two-server PEKS may still suffer from the inefficiency as the keyword searching is now separately processed by two servers.

In this work, we aim at designing a more practical treatment to address this security issue. Moreover, we are interested in building a system that works transparently with any existing PEKS system. That is, the system will be backward-compatible and make no modification on the implementation details of the underlying PEKS system.

B. Our Contributions

The contributions of this paper are four-fold. First, we formalize a new PEKS system named *Server-Aided Public Key Encryption with Keyword Search* (SA-PEKS) to address the security vulnerability against (inside) off-line KGA in existing PEKS systems. Our proposed solution can work transparently with any existing PEKS system and hence is much more applicable in practice. Secondly, we present a generic construction of SA-PEKS scheme with formal security analysis. Precisely, we propose a universal transformation from any PEKS scheme to an SA-PEKS scheme by utilizing a deterministic blind signature. Thirdly, to illustrate the feasibility of the proposed generic transformation, an instantiation of the SA-PEKS scheme is presented in this paper. We instantiate the scheme from the FDH-RSA blind signature and the PEKS scheme proposed by Boneh *et al.* [3]. Lastly, we present the implementation of our solution and analyze its performance in experiments. Particularly, we show how to securely implement the client-KS protocol with a rate-limiting mechanism against on-line KGA.

1) Related Work:

a) Traditional PEKS: Following Boneh et al.'s seminal work [3], Abdalla *et al.* [8] formalized anonymous IBE (AIBE) and presented a generic construction of searchable encryption from AIBE. In order to construct a PEKS secure in the standard model, Khader [9] proposed a scheme based on the k-resilient IBE. In [10], an interesting primitive called searchable public-key ciphertexts with hidden structures (SPCHS) was proposed for efficient keyword search without sacrificing semantic security of the encrypted keywords.

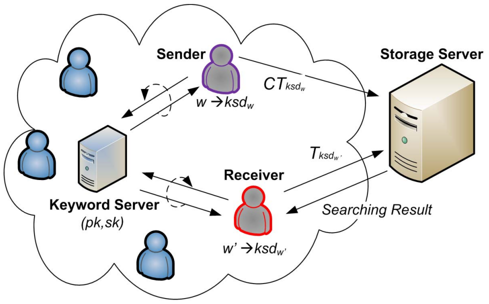


Fig. 1. System Model of Server-Aided PEKS.

b) Secure channel-free PEKS: Baek *et al.* [11] proposed a new PEKS scheme, which is referred to as a secure channel-free PEKS (SCF-PEKS). Rhee *et al.* [12] later enhanced Baek *et al.*'s security model [11] for SCF-PEKS where the attacker is allowed to obtain the relationship between the non-challenge ciphertexts and the trapdoor. They also presented an SCF-PEKS scheme secure under the enhanced security model in the random oracle model.

c) Against outside KGA: Byun *et al.* [13] introduced the off-line keyword guessing attack against PEKS as keywords are chosen from a much smaller space than passwords and users usually use well-known keywords for searching documents. Inspired by the work of Byun *et al.* [13], Yau *et al.* [14] demonstrated that outside adversaries that capture the trapdoors sent in a public channel can reveal the encrypted keywords through off-line keyword guessing attacks and they also showed off-line keyword guessing attacks against the (SCF-)PEKS schemes in [11] and [15]. The first PEKS scheme secure against outside keyword guessing attacks was proposed by Rhee *et al.* [16].

II. Server-Aided PEKS

A. An Overview

SA-PEKS is motivated by the observation that the off-line KGA can be dealt with by employing a semi-trusted third party, namely *Keyword Server* (KS) which is separated from the *Storage Server* (SS), as shown in Figure.1. Roughly speaking, in an SA-PEKS system, the KS owns the public/secret key pair (pk, sk). Users authenticate themselves to the KS and are provisioned with per-user credentials. Different from the PEKS framework where the PEKS ciphertext and the trapdoor are derived from the original keyword directly, the user needs to interact with the KS in an authenticated way to obtain the pre-processed keyword, namely *KS-derived keyword*, before the generation of the PEKS ciphertext and the trapdoor. More specifically, given an original keyword w, the sender has to access the KS through authentication and run an interactive protocol with the KS. At the end of the protocol execution, the sender gets the corresponding KS-derived keyword of w as ksd_w. The sender then generates the PEKS ciphertext by regarding the KS-derived keyword ksd_w as the final keyword. Similarly, taking as input a specified keyword w', the receiver runs the interactive protocol with the KS to obtain the KS-derived keyword $ksd_{w'}$ and then generates the corresponding trapdoor. It is required that the derivation algorithm from original keyword to KS-derived keyword should be deterministic, otherwise the SA-PEKS cannot work correctly. That is, if $w = w'$, then we have that $ksd_w = ksd_{w'}$. We can see that in this way, the generation of PEKS ciphertexts and trapdoors turns to be in an on-line manner (through protocol) and hence the security against the off-line KGA can be obtained. Moreover, the KS can function as a single point of control for implementing rate-limiting measures to reduce the on-line KGA rate.

B. Formal Definition

Definition 4 (Server-Aided PEKS): An SA-PEKS *scheme is defined by the six-tuple* ($\mathsf{SA\text{-}KeyGen}_{\mathcal{KS}}$, $\mathsf{SA\text{-}KeyGen}_{\mathcal{R}}$, SA-KSD, SA-PEKS, SA-Trapdoor, SA-Test) *as follows.*

$\mathsf{SA\text{-}KeyGen}_{\mathcal{KS}}(\lambda)$. *Taking as input the security parameter λ, outputs the public/private key pair of the KS as (pk_{ks}, sk_{ks}).*

$\mathsf{SA\text{-}KeyGen}_{\mathcal{R}}(\lambda)$. *Taking as input parameter λ, it outputs the public/private key pair of the receiver as (pk_R, sk_R).*

$\mathsf{SA\text{-}KSD}(pk_{ks}, sk_{ks}, w)$. *Taking as input the key pair of the KS and the keyword w, it returns the KS-derived keyword ksd_w.*

$\mathsf{SA\text{-}PEKS}(pk_R, ksd_w)$. *Taking as input the public key pk_R of the receiver and the KS-derived keyword ksd_w, the sender outputs the PEKS ciphertext of w as CT_{ksd_w}.*

$\mathsf{SA\text{-}Trapdoor}(sk_R, ksd_{w'})$. *Taking as input the secret key sk_R of the receiver and the KS-derived keyword $ksd_{w'}$, the receiver outputs the the trapdoor as $T_{ksd_{w'}}$.*

$\mathsf{SA\text{-}Test}(pk_R, CT_{ksd_w}, T_{ksd_{w'}})$. *Taking as input the public key pk_R, the PEKS ciphertext CT_{ksd_w} and the trapdoor $T_{ksd_{w'}}$, the SS outputs $True$ if $w = w'$; otherwise outputs $False$.*

Correctness: It is required that for any two keywords w, w', $ksd_w \leftarrow \mathsf{SA\text{-}KSD}(pk_{ks}, sk_{ks}, w), ksd_{w'} \leftarrow \mathsf{SA\text{-}KSD}(pk_{ks}, sk_{ks}, w'), CT_{ksd_w} \leftarrow \mathsf{SA\text{-}PEKS}(pk_R, ksd_w)$ and $T_{ksd_{w'}} \leftarrow \mathsf{SA\text{-}Trapdoor}(sk_R, ksd_{w'})$, we have that $\mathsf{SA\text{-}Test}(pk_R, CT_{ksd_w}, T_{ksd_{w'}}) = True$ if $w = w'$.

Remark 1: The algorithm SA-KSD is an interactive protocol between the user (sender/receiver) and the KS. Both the KS and the user take as input the public information pk_{ks}. The private input of the KS is sk_{ks} while that for the user is the original keyword. The KS and the user engage in the KS-derived keyword issuing protocol and stop in polynomial time. When the protocol completes, the private output of the user contains the KS-derived keyword. Without loss of generality, we assume that the user can verify the validity of the KS-derived keyword by using the public information and hence the algorithm SA-KSD always outputs the KS-derived keyword upon the successful completion of the interactive protocol.

C. Security Models

In this subsection, we define the security models for the SA-PEKS in terms of the adversarial SS, the honest but curious KS and adversarial users respectively.

1) Adversarial Storage Server (SS): Here we propose a new notion, namely *Semantic-Security against Chosen Keyword Guessing Attack* (SS-CKGA) for the SA-PEKS. Similar to the notion of SS-CKA in PEKS, SS-CKGA guarantees that the PEKS ciphertext in the SA-PEKS does not reveal any information about the underlying keyword. The difference between the SS-CKGA and SS-CKA is that the adversary against the SA-PEKS is allowed to obtain the matching trapdoor of the challenge PEKS ciphertext.

Definition 5 (SS-CKGA): The SS-CKGA *game is as follows.*

Setup. *The challenger generates key pairs (pk_R, sk_R), (pk_{ks}, sk_{ks}) and sends (pk_R, pk_{ks}) to the attacker.*

Query-I. *The attacker can adaptively query the challenger for the trapdoor and PEKS ciphertext of any keyword.*

Challenge. *The attacker sends the challenger two keywords w_0, w_1. The restriction here is that none of w_0 nor w_1 has been queried in the* Query-I. *The challenger picks $b \overset{\$}{\leftarrow} \{0, 1\}$ and generates $ksd_{w_b} \leftarrow \mathsf{SA\text{-}KSD}(pk_{ks}, sk_{ks}, w_b)$, $CT^* \leftarrow \mathsf{SA\text{-}PEKS}(pk_R, ksd_{w_b})$, $T^* \leftarrow \mathsf{SA\text{-}Trapdoor}(sk_R, ksd_{w_b})$.*

The challenger then sends (CT^*, T^*) to the attacker.

Query-II. *The attacker can continue the query for the trapdoor and PEKS ciphertext of any keyword of its choice except of the challenge keywords w_0, w_1.*

Output. *Finally, the attacker outputs its guess $b' \in \{0, 1\}$ on b and wins the game if $b = b'$.*

We refer to such an adversary $\mathcal{A}$ in the above game as an SS-CKGA adversary and define its advantage as $\mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}(\lambda) = \Pr[b = b'] - 1/2$.

2) Honest but Curious Keyword Server (KS): It is required that the protocol cannot reveal any information about the private input of the user to the KS or other outside attackers. Formally, we define the notion of *Indistinguishability against Chosen Keyword Attack* (IND-CKA) as follows.

Definition 6 (IND-CKA): The IND-CKA *game is defined as,*

Setup. *The challenger runs the algorithm* KeyGen*(λ) and sends the attacker the key pair (pk_{ks}, sk_{ks}). The attacker then sends the challenger two keywords w_0, w_1.*

Challenge. *The challenger picks $b \overset{\$}{\leftarrow} \{0, 1\}$, runs the KS-derived keyword issuing protocol with the attacker by taking as input the keyword w_b.*

Output. *After the protocol ends, the attacker outputs its guess $b' \in \{0, 1\}$ on b and wins the game if $b = b'$.*

We refer to such an adversary $\mathcal{A}$ in the above game as an IND-CKA adversary and define its advantage as $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathcal{KS},\mathcal{A}}(\lambda) = \Pr[b = b'] - 1/2$.

3) Adversarial Users: It is a requirement that only the KS can generate the correct KS-derived keywords, otherwise the security of SA-PEKS falls to that of original PEKS, i.e., being insecure against off-line KGA. Also, we should prevent an adversarial user from generating the KS-derived keyword based on the previous KS-derived keywords obtained from the KS. Therefore, to best capture such a security requirement, we define the notion of *One-More-Unforgeability under Chosen Keyword Attack* (OMU-CKA) as follows.

Definition 7 (OMU-CKA): The OMU-CKA *game is defined as,*

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

Setup. *The challenger runs algorithm* $\mathsf{KeyGen}(\lambda)$ *to obtain key pair* (pk_{ks}, sk_{ks}). *The attacker is given* pk_{ks}.

$\mathsf{KSD\text{-}Query}$. *The attacker can adaptively query the challenger for the KS-derived keyword for at most* q_k *distinct original keywords of his choice* $w_1, w_2, ..., w_{q_k}$.

Output. *Finally, the attacker outputs* $q_k + 1$ *pairs* $\{w_i, ksd_{w_i}\}_{i \in [1, q_k+1]}$ *and wins the game if (1)* $w_i \neq w_j$, *for any* $i, j \in [1, q_k + 1]$ *where* $i \neq j$, *and (2)* ksd_{w_i} *is a valid KS-derived keyword of* w_i *for any* $i \in [1, q_k + 1]$.

We refer to such an adversary $\mathcal{A}$ in the above game as an OMU-CKA adversary and define its advantage $\mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}}(\lambda)$ to be the probability that $\mathcal{A}$ wins in the above game.

Based on the security models defined above, we give the following security definition for an SA-PEKS scheme.

Definition 8 (Secure SA-PEKS): We say that an SA-PEKS is secure if for any polynomial time attacker $\mathcal{A}_i$ $(i = 1, 2, 3)$, we have that $\mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}_1}(\lambda)$, $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathcal{KS},\mathcal{A}_2}(\lambda)$ and $\mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}_3}(\lambda)$ are all negligible functions of the security parameter λ.

III. PEKS-TO-SA-PEKS TRANSFORMATION

A. Blind Signature

Before introducing the universal transformation, we briefly review the definition of blind signature. Blind signature was introduced by Chaum [17].

1) Syntax: Formally, a blind signature scheme is an interactive scheme that consists of a tuple of algorithms $(\mathsf{Kg}, \mathsf{Sign}, \mathsf{User}, \mathsf{Vf})$. Suppose that the system security parameter is λ. The signer generates a key pair via the key generation algorithm $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(\lambda)$. To obtain a signature on a message $m \in \{0, 1\}^*$, the user and signer engage in an interactive signing protocol dictated by the algorithms $\mathsf{User}(pk, m)$ and $\mathsf{Sign}(sk)$. After the protocol completes, the User algorithm locally outputs a signature σ_m on m. To verify the validity of a signature σ_m, the verification algorithm Vf takes as input pk, m and σ_m, outputs $True$ if the signature is valid and $False$ otherwise. A blind signature scheme has correctness if $\mathsf{Vf}(pk, m, \sigma_m) = True$ for any $(pk, sk) \xleftarrow{\$} \mathsf{Kg}(\lambda)$, any message m and any signature σ_m output by $\mathsf{User}(pk, m)$ after interacting with $\mathsf{Sign}(sk)$.

A blind signature is *deterministic* if for each public key pk and each message m, there exists only one signature σ_m such that $\mathsf{Vf}(pk, m, \sigma) = True$.

2) Security: The security of blind signature is twofold: *unforgeability* and *blindness*. We say a blind scheme is *one-more-unforgeable* if any polynomial time adversary that queries the signing oracle with q_s distinct messages can only forge $q_s + 1$ valid message/signature pairs with negligible probability. Another notion, namely *blindness*, requires that the signer cannot tell apart the message it is signing. To be more precise, the blindness condition says that it should be infeasible for a malicious signer to decide which of the two messages has been signed first in two executions with an honest user.

B. A Universal Transformation

In this subsection, we show a universal transformation from PEKS to SA-PEKS. Given a deterministic blind signature scheme $\mathcal{BS} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{User}, \mathsf{Vf})$ and a PEKS scheme $\mathcal{PEKS} = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Trapdoor}, \mathsf{Test})$, the resulting SA-PEKS scheme is as follows. Let $\widehat{H} : \{0, 1\}^* \times \{0, 1\}^* \to \{0, 1\}^*$ be a cryptographic collision-resistant hash function.

$\mathsf{SA\text{-}KeyGen}_{\mathcal{KS}}(\lambda)$. Take as input the security parameter λ and output the public/private key pair of the KS by running the algorithm Kg of $\mathcal{BS}$ as $(pk_{ks}, sk_{ks}) \xleftarrow{\$} \mathsf{Kg}(\lambda)$.

$\mathsf{SA\text{-}KeyGen}_{\mathcal{R}}(\lambda)$. Take as input the parameter λ and output the key pair of the receiver by running the algorithm KeyGen of $\mathcal{PEKS}$ as $(pk_R, sk_R) \xleftarrow{\$} \mathsf{KeyGen}(\lambda)$.

$\mathsf{SA\text{-}KSD}(pk_{ks}, sk_{ks}, w)$. Take as input the key pair of the KS and the keyword w, the user (sender/receiver) runs the algorithm $\mathsf{User}(pk_{ks}, w)$ and the KS runs the algorithm $\mathsf{Sign}(sk_{ks})$ of $\mathcal{BS}$ in an interactive signing protocol to obtain the valid signature σ_w of w. The user finally obtains the KS-derived keyword as $ksd_w = \widehat{H}(w, \sigma_w)$.

$\mathsf{SA\text{-}PEKS}(pk_R, ksd_w)$. Take as input the public key pk_R and the KS-derived keyword ksd_w, the sender runs the algorithm PEKS of $\mathcal{PEKS}$ as, $CT_{ksd_w} \leftarrow \mathsf{PEKS}(pk_R, ksd_w)$. It then outputs CT_{ksd_w} as the PEKS ciphertext of w.

$\mathsf{SA\text{-}Trapdoor}(sk_R, ksd_{w'})$. Take as input the secret key sk_R of the receiver and the KS-derived keyword $ksd_{w'}$, the receiver runs $\mathsf{Trapdoor}$ of $\mathcal{PEKS}$ as, $T_{ksd_{w'}} \leftarrow \mathsf{Trapdoor}(sk_R, ksd_{w'})$. It then outputs $T_{ksd_{w'}}$ as the trapdoor of w'.

$\mathsf{SA\text{-}Test}(pk_R, CT_{ksd_w}, T_{ksd_{w'}})$. Take as input the public key pk_R, the PEKS ciphertext CT_{ksd_w} and the trapdoor $T_{ksd_{w'}}$, the SS runs the algorithm Test of $\mathcal{PEKS}$ as, $True/False \leftarrow \mathsf{PEKS}(pk_R, CT_{ksd_w}, T_{ksd_{w'}})$. It then outputs $True/False$ as the testing result.

1) Correctness Analysis: One can see that the correctness condition of this construction holds due to the collision-resistant hash function $\widehat{H}$, the deterministic scheme $\mathcal{BS}$ and the correctness of $\mathcal{PEKS}$. To be more precise, for any keywords w, w', we have that $\sigma_w = \sigma'_w$ and hence $ksd_w = ksd_{w'}$ if $w = w'$; otherwise, we have that $ksd_w \neq ksd_{w'}$. Therefore, due to the correctness of $\mathcal{PEKS}$, we have that $True \leftarrow \mathsf{PEKS}(pk_R, CT_{ksd_w}, T_{ksd_{w'}})$ when $w = w'$; otherwise, $False \leftarrow \mathsf{PEKS}(pk_R, CT_{ksd_w}, T_{ksd_{w'}})$.

C. Security Analysis

1) SS-CKGA Security: Formally, the SS-CKGA security of the above SA-PEKS scheme $\mathcal{SA\text{-}PEKS}$ is guaranteed by the following theorem.

> *Theorem 1: Let hash function* $\widehat{H}$ *be a random oracle. Suppose that there exits a polynomial-time adversary* $\mathcal{A}$ *that can break the SS-CKGA security of* $\mathcal{SA\text{-}PEKS}$ *with advantage* $\mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}$, *then there exists a polynomial-time adversary* $\mathcal{B}$ *that can break the one-more unforgeability of the underlying signature scheme* $\mathcal{BS}$ *with advantage at least* $\mathsf{Adv}^{\mathsf{OMU}}_{\mathcal{BS},\mathcal{B}} \geq 1/q_{\widehat{H}} \cdot \mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}$, *where* $q_{\widehat{H}}$ *is the number of queries to* $\widehat{H}$.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

Proof: We prove the theorem by constructing an algorithm $\mathcal{B}$ who simulates the challenger in the SS-CKGA model to play the game with $\mathcal{A}$. The goal of $\mathcal{B}$ is to break the *one-more-unforgeability* security of the scheme $\mathcal{BS}$. Suppose that $\mathcal{B}$ is given the public key pk_{ks} of $\mathcal{BS}$. Then $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

In the Setup stage, $\mathcal{B}$ runs the algorithm KeyGen to generate the key pair (pk_R, sk_R) and gives pk_R, pk_{ks} to the adversary $\mathcal{A}$. In the Query-I stage, when $\mathcal{A}$ queries a keyword w for the PEKS ciphertext (or the trapdoor), $\mathcal{B}$ first interacts with the signing oracle to obtain the signature of w and accesses to the random oracle $\widehat{H}$ for the KS-derived keyword ksd_w. $\mathcal{B}$ then generates the PEKS ciphertext CT_{ksd_w} (or the trapdoor T_{ksd_w}) using (pk_R, sk_R) and returns the result to $\mathcal{A}$. In the Challenge stage, upon receiving two challenge keywords w_0, w_1 from $\mathcal{A}$, instead of querying w_b to the signing oracle for the signature σ_{w_b}, $\mathcal{B}$ just picks randomly r and generates the challenge PEKS ciphertext and trapdoor (CT^*, T^*) of w_b by regarding r as the signature of w_b. $\mathcal{A}$ then sends (CT^*, T^*) to $\mathcal{A}$. In the Query-II stage, $\mathcal{B}$ simulates as that in Query-I stage. In the Ouput stage, after $\mathcal{A}$ outputs its guess b' on b, $\mathcal{B}$ picks an input-element from the random oracle $\widehat{H}$ randomly and outputs it as its forgery signature.

Let Q be the event that $\mathcal{A}$ queried (w_b, σ_{w_b}) to the random oracle $\widehat{H}$. Then the advantage of $\mathcal{A}$ wins in the above game is $\mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}} = \Pr[b' = b|Q] + \Pr[b' = b|\overline{Q}] - 1/2$. One can note that in the above simulation, if event Q does not happen, that is, $\mathcal{A}$ did not ever query $\widehat{H}$ with (w_b, σ_{w_b}), then the above game is identical to the original SS-CKGA game from the view of $\mathcal{A}$ due to the property of random oracle $\widehat{H}$. However, the probability of $\mathcal{A}$ wins in this game under this case is at most $1/2$ since (CT^*, T^*) is independent of b, i.e. $\Pr[b' = b|\overline{Q}] = 1/2$. Therefore, we have that $\Pr[b' = b|Q] = \mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}$, which means that the event Q happens with probability $\mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}$. Therefore, $\mathcal{B}$ can successfully forgery a signature as (w_b, σ_{w_b}) with advantage $\mathsf{Adv}^{\mathsf{OMU}}_{\mathcal{BS},\mathcal{B}} \geq 1/q_{\widehat{H}} \cdot \mathsf{Adv}^{\mathsf{SS\text{-}CKGA}}_{\mathcal{SS},\mathcal{A}}$. □

2) IND-CKA Security: As for the IND-CKA security which guarantees that except the user no other entity can learn any information about the private input (keyword) of the user, we have the following theorem.

Theorem 2: If there exits a polynomial-time adversary $\mathcal{A}$ that can break the IND-CKA security of the above scheme $\mathcal{SA}$-$\mathcal{PEKS}$ with advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathcal{KS},\mathcal{A}}$, then there exists a polynomial-time adversary $\mathcal{B}$ that can break the blindness security of the underlying signature scheme $\mathcal{BS}$ with advantage $\mathsf{Adv}^{\mathsf{Blindness}}_{\mathcal{BS},\mathcal{B}} \geq \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathcal{KS},\mathcal{A}}$.

Proof: We prove the theorem above by constructing an algorithm $\mathcal{B}$ who runs the adversary $\mathcal{A}$ as a subroutine to break the security of *blindness* of the underlying scheme $\mathcal{BS}$ as follows. Suppose the challenger in the *blindness* security attack game against the scheme $\mathcal{BS}$ is $\mathcal{C}$.

In the Setup stage, $\mathcal{B}$ receives the public/private key pair (pk, sk) from $\mathcal{C}$, sends the key pair to $\mathcal{A}$ and then receives the challenge keywords (w_0, w_1) from $\mathcal{A}$. $\mathcal{B}$ then forwards (w_0, w_1) as the challenge message to $\mathcal{C}$. In the Challenge stage, $\mathcal{B}$ simulates as the adversarial signer from the view of $\mathcal{C}$ and as the challenger of the IND-CKA game against the scheme $\mathcal{SA}$-$\mathcal{PEKS}$ from the view of $\mathcal{A}$. Once $\mathcal{C}$ starts the first execution of the signing protocol, $\mathcal{B}$ starts the KS-derived keyword issuing protocol with $\mathcal{A}$ and forwards the message between $\mathcal{A}$ and $\mathcal{C}$. During the second execution of the protocol, $\mathcal{B}$ interacts with $\mathcal{C}$ honestly using the public/private key pair (pk, sk). In the Ouput stage, after $\mathcal{A}$ outputs its guess b', $\mathcal{B}$ outputs b' as its guess to $\mathcal{C}$.

It is easy to see that the above simulation is indistinguishability from the IND-CKA game from the view of $\mathcal{A}$. Therefore, we have that $\mathsf{Adv}^{\mathsf{Blindness}}_{\mathcal{BS},\mathcal{B}} \geq \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathcal{KS},\mathcal{A}}$, □

3) OMU-CKA Security: Here we discuss the OMU-CKA security of our universal transformation. This notion guarantees that a user cannot forge a new KS-derived keyword without the help of the KS even if it has seen many KS-derived keywords before. Formally, we have the following theorem.

Theorem 3: If there exits a polynomial-time adversary $\mathcal{A}$ that can break the OMU-CKA security of the above scheme $\mathcal{SA}$-$\mathcal{PEKS}$ with advantage $\mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}}$, then there exists a polynomial-time adversary $\mathcal{B}$ that can break the one-more-unforgeability security of the underlying signature scheme $\mathcal{BS}$ with advantage at least $\mathsf{Adv}^{\mathsf{OMU}}_{\mathcal{BS},\mathcal{B}} \geq \mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}}$.

Proof: We give the proof of the theorem above by constructing an algorithm $\mathcal{B}$ who invokes the adversary $\mathcal{A}$ to break the *one-more-unforgeability* security of the scheme $\mathcal{BS}$ as follows.

In the Setup stage, $\mathcal{B}$ receives the public key pk_{ks} from the signing oracle and sends pk_{ks} to $\mathcal{A}$. In the KSD-Query stage, upon receiving a queried keyword w, $\mathcal{B}$ queries w to the signing oracle to obtain the signature σ_w, returns the hash value of the returned signature as $\widehat{H}(w, \sigma_w)$ to $\mathcal{A}$ as the KS-derived keyword of w. In the Ouput stage, if $\mathcal{A}$ outputs $q_k + 1$ valid pairs $\{(w_i, ksd_{w_i})\}_{1 \leq i \leq q_k+1}$ where q_k is the KS-derived keyword query number. Then for each i, $\mathcal{B}$ looks up w_i in the hash query record for the corresponding signature σ_i and outputs $\{(w_i, \sigma_i)\}_{1 \leq i \leq q_k+1}$ as the $q_k + 1$ valid message/signature pairs as its forgery signatures. Otherwise, $\mathcal{B}$ aborts.

One can see that the simulation above by $\mathcal{B}$ is indistinguishable from the original OMU-CKA game from the view of the adversary $\mathcal{A}$, therefore, $\mathcal{A}$ can successfully output $q_k + 1$ valid keyword/KS-derived keywords pairs with the advantage $\mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}}(\lambda)$. That is, $\mathcal{B}$ can break the *one-more-unforgeability* security of $\mathcal{BS}$ with advantage at least $\mathsf{Adv}^{\mathsf{OMU}}_{\mathcal{BS},\mathcal{B}} \geq \mathsf{Adv}^{\mathsf{OMU\text{-}CKA}}_{\mathcal{U},\mathcal{A}}$. □

Based on the above, we have the following observation.

Theorem 4: The universal transformation above results in a secure SA-PEKS scheme if the underlying blind signature is secure in terms of one-more-unforgeability and blindness.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

4) Further Discussions on the Multi-Tiered Security: Ideally, we hope that the adversary does not have authorized access to the KS. This means that the adversary can launch neither off-line nor on-line KGA. However, in reality, the adversary (including the adversarial SS) may have remote access to the KS. In this case, the resulting SA-PEKS still remains SS-CKA secure as long as the underlying PEKS is SS-CKA secure. However, we should note that even in this case, the adversarial SS cannot efficiently launch the KGA since it needs to query the signature of each guessing keyword in an on-line manner through the protocol and hence the brute-force attack will be rendered less effective. As will shown in Section 6.1, with an effective rate-limiting mechanism, the on-line KGA can be slowed down significantly.

IV. Instantiations of SA-PEKS

A. Underlying Schemes

Following the above universal transformation, here we show an instantiation of the proposed SA-PEKS scheme based on the FDH-RSA blind signature [19] and the PEKS scheme (denoted by BCOP-PEKS) proposed by Boneh *et al.* in [3]. We start with the introduction of the two building blocks.

1) FDH-RSA: The signer has public key $N, \widehat{e}$ and private key N, d where $\widehat{e}d \equiv 1 \mod \phi(N)$, modulus N is the product of two distinct primes of roughly equal length. The user uses a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ to hash the message m to an element of $\mathbb{Z}_N$ and then blinds result with a random group element $r \overset{\$}{\leftarrow} \mathbb{Z}_N^*$. The resulting blinded hash, denoted $\overline{m}$ is then sent to the signer. The signer signs $\overline{m}$ with its private key d by computing $\overline{\sigma} \leftarrow (\overline{m})^d \mod N$ and sends back $\overline{\sigma}$. The user then derives the signature by removing the blinding element through $\sigma \leftarrow r^{-1}\overline{\sigma} \mod N$. The correctness can be obtained due to the fact that $\overline{\sigma} = (\overline{m})^d \mod N = (r^{\widehat{e}}H(m))^d \mod N = rH(m)^d \mod N$ and hence $\sigma = r^{-1}\overline{\sigma} \mod N$.

We can see that the *blindness* security of the above FDH-RSA is guaranteed by the one-time element chosen randomly to blind the signed message. As for the unforgeability security, based on the result from [19], we have the conclusion that the FDH-RSA blind signature scheme is polynomially-secure against one-more forgery if the *RSA known-target inversion* problem is hard. More details can be found in [19].

2) BCOP-PEKS: Here, we show the PEKS scheme proposed in [3]. This scheme is based on a variant of the *Computation Diffie-Hellman Problem*. Let $\mathbb{G}_1, \mathbb{G}_T$ be two multiplicative groups with the same prime order p. Let g be the generator of $\mathbb{G}_1$ and I be the identity element of $\mathbb{G}_T$. A symmetric bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ such that $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$. It is worth noting that e can be efficiently computed and $e(g, g) \neq I$. Then the non-interactive searchable encryption scheme works as follows.

KeyGen. The input security parameter determines the size p of the groups $\mathbb{G}_1, \mathbb{G}_T$. The algorithm then picks a random $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, a generate g of $\mathbb{G}_1$ and choose two hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_T \to \{0,1\}^{\log p}$. Then it outputs $pk_R = (g, h = g^\alpha, H_1, H_2), sk_R = \alpha$.

PEKS. For a keyword w, pick a random $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute $t = e(H_1(w), h^r)$. The sender then outputs the PEKS ciphertext as $CT_w = (g^r, H_2(t))$.

Trapdoor. For a keyword w', The receiver outputs the trapdoor as $T_{w'} = H_1(w')^\alpha \in \mathbb{G}_1$.

Test. The server takes as input $CT_w = (A, B)$ and $T_{w'}$, if $H_2(e(T_{w'}, A)) = B$, outputs $True$, $False$ otherwise.

The correctness of the PEKS scheme above can be easily obtained. In terms of security, the scheme is secure against the SS-CKA [3] but insecure against the off-line KGA. Actually, the off-line KGA against the BCOP-PEKS scheme can be launched in a much simpler way. Given a trapdoor $T_{w^*} = H_1(w^*)^\alpha$, the attacker can easily test whether its guessing keyword w is the underlying keyword of T_{w^*} by checking $e(T_{w^*}, g) \overset{?}{=} e(H_1(w), h)$.

B. Resulting SA-PEKS

Here we show the resulting SA-PEKS derived from the FDH-RSA and the BCOP-PEKS schemes. The details are described in Fig.2. Note that the KS-derived keyword issuing protocol in our scheme requires that $N < \widehat{e}$ should be verified by the user. This is to avoid that the KS may generate the keys dishonestly in order to learn some information about the keyword. This condition ensures that $\gcd(\phi(N), \widehat{e}) = 1$ even if N is maliciously generated and thus ensures that the map $f_{\widehat{e}} : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, defined by $f_{\widehat{e}}(x) = x^{\widehat{e}} \mod N$ for all $x \in \mathbb{Z}_N^*$, is a permutation on $\mathbb{Z}_N^*$. Since $f_{\widehat{e}}$ is a permutation and the user can verify the validity of the signature, even a malicious KS cannot force the output of signature to be a fixed value.

It is easy to see that for any two keywords w, w', if $w = w'$, then $ksd_w = \widehat{H}(w, \sigma_w) = \widehat{H}(w, H(w)^d) = \widehat{H}(w', H(w')^d) = ksd_{w'}$. Therefore, as for the corresponding PEKS ciphertext $CT_{ksd_w} = (A, B)$ and the trapdoor $T_{ksd_{w'}}$, we have that $H_2(e(T_{ksd_{w'}}, A)) = H_2(e(H_1(ksd_{w'}), h^r)) = B$. Otherwise, $H_2(e(T_{ksd_{w'}}, A)) \neq B$ as $ksd_w \neq ksd_{w'}$.

1) Security Analysis: The security of the resulting SA-PEKS scheme can be easily obtained based on ***Theorem 4***, as the FDH-RSA is *one-more-unforgeable* and of *blindness*. Formally, we have the following collusion.

Corollary 1. The above concrete SA-PEKS scheme is secure.

V. Implementation and Performance

A. The Client-KS Protocol

Motivated by the work in [20], we show a protocol for client-KS interaction and the rate-limiting strategies which limit client queries to slow down on-line keyword guessing attack. Our design goal is to give a low-latency protocol to avoid performance degrading. The proposed protocol relies on a CA providing the KS and each client with a unique verifiable TLS certificates As shown in Fig. 3, the execution of protocol consists of the Mutual Authentication (MA) phase and the Query-Response (QR) phase, of which the first one is over HTTP while the later one is over UDP.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

Let $\mathbb{G}_1, \mathbb{G}_T$ be two groups with prime order p and the bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. The public RSA-exponent $\widehat{e}$ is fixed as part of the scheme. Let $\widehat{H} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be a cryptographic collision-resistant hash function.

SA-KeyGen$_{\mathcal{KS}}(\lambda)$. The algorithm runs Kg$(\lambda, \widehat{e})$ to get N, d such that $\widehat{e}d \equiv 1 \mod \phi(N)$ and $N < \widehat{e}$. It chooses hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$, and then outputs $pk_{ks} = (N, \widehat{e}, H, \widehat{H}), sk_{ks} = (N, d)$.

SA-KeyGen$_{\mathcal{R}}(\lambda)$. The algorithm takes as input the security parameter, determines the size p of the groups $\mathbb{G}_1, \mathbb{G}_T$ and picks a random $\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, a generate g of $\mathbb{G}_1$, computes $h = g^\alpha$ and chooses hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_T \leftarrow \{0,1\}^{\log p}$. it then outputs $pk_R = (g, h, H_1, H_2), sk_R = \alpha$.

SA-KSD. For a keyword w, the algorithm runs the KS-derived keyword issuing protocol to obtain the valid signature σ_w of w as follows.

User $(N, \widehat{e}, H, w)$ **KS**(N, d)

If $\widehat{e} \leq N$ then abort

$r \overset{\$}{\leftarrow} \mathbb{Z}_N^*$

$\overline{w} \leftarrow r^{\widehat{e}} \cdot H(w) \mod N$

$\xrightarrow{\overline{w}}$

$\overline{\sigma} \leftarrow (\overline{w})^d \mod N$

$\xleftarrow{\overline{\sigma}}$

$\sigma_w \leftarrow r^{-1}\overline{\sigma} \mod N$

If $\sigma_w^{\widehat{e}} \mod N \neq H(w)$ then abort

Else return σ_w

The user (sender/receiver) then computes $ksd_w = \widehat{H}(w, \sigma_w) = \widehat{H}(w, H(w)^d)$, and outputs ksd_w as the KS-derived keyword of w.

SA-PEKS. For a KS-derived keyword ksd_w, the sender picks a random $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, computes $t = e(H_1(ksd_w), h^r), CT_{ksd_w} = (g^r, H_2(t))$. The algorithm outputs CT_{ksd_w} as the PEKS ciphertext of ksd_w.

SA-Trapdoor. For a KS-derived keyword $ksd_{w'}$, the receiver computes $T_{ksd_{w'}} = H_1(ksd_{w'})^\alpha$, and outputs $T_{ksd_{w'}}$ as the trapdoor of $ksd_{w'}$.

SA-Test. The SS takes as input $CT_{ksd_w} = (A, B)$ and $T_{ksd_{w'}}$, and checks $H_2(e(T_{ksd_{w'}}, A)) \overset{?}{=} B$. If yes, output $True$, else output $False$.

Fig. 2. An SA-PEKS scheme from FDH-RSA and BCOP-PEKS scheme.

1) Phase I (Mutual Authentication): The MA procedure starts with a TLS handshake with mutual authentication, initiated by a client. The KS responds immediately with the verification key pk of the underlying blind signature scheme, a hash function H (by default SHA-256), a random session identifier $S \in \{0,1\}^{128}$, and a random session key $K_S \in \{0,1\}^k$. The KS then initializes and records a sequence number with this session as $N_S = 0$. The client stores pk, S, K_S and also initializes a sequence number $N_C = 0$.

2) Phase II (Query-Response): In the QR phase, the client first generates a blinded value of a keyword as W_R, increases the recorded sequence number $N_C \leftarrow N_C + 1$, and then computes a MAC tag using the KS's session key K_S, as $T \leftarrow \mathsf{HMAC[H]}(K_S, S||N_C||W_R)$. The client then sends $S||N_C||W_R||T$ to the KS in a UDP packet. Upon receiving the query information, the KS first checks that $N_C \geq N_S$ and verifies the correctness of the MAC T. If the verification fails, then the KS drops the packet without any further response. Otherwise, it would signs the blinded keyword and returns the signature σ to the client.

3) Per-Client Rate-Limiting Mechanism Against On-Line KGA: We explore the so-called *exponential delay mechanism* to achieve a balancing between defence against the on-line KGA and the latency of a KS request. For the first query, the KS performs the response with an initial small delay t_I, and the delay time is doubled after each query from the same client. The doubling then stops at an upper limit t_U. The KS maintains synchronized epochs and an active client list. It checks the status of active clients after each epoch. The delay would be reset to the initial value if the client makes no queries during an entire epoch. It would also drop any new query from the active client who is in the list and awaiting responses. We remark that our strategy would not bring much communication delay for the normal user who usually performs the query at a low rate.

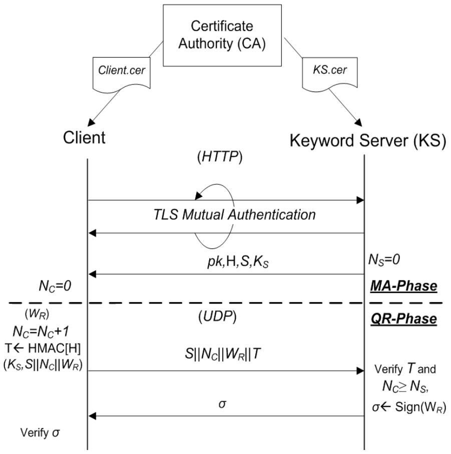


Fig. 3. The Client-KS Protocol in SA-PEKS.

4) Protocol Security: Attackers can attempt to eavesdrop and even tamper with the communications between clients and

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

TABLE I
LATENCY OF PROTOCOL UNDER DIFFERENT LOAD

Operation	**Latency (ms)**
Ping (1 Round-Trip Time)	96 ±02
MA Phase	312 ± 48
QR Phase (1000 q/s)	103 ± 32
QR Phase (2000 q/s)	134 ±43
QR Phase (3000 q/s)	157 ±40
QR Phase(4000 q/s)	193 ±46
QR Phase (5000 q/s)	252 ±51
QR Phase (6000 q/s)	327 ±49

the KS. In the protocol, due to the mutual authentication TLS handshake in the session initialization, no adversary can start a session pretending to be a valid client. Moreover, without the session key K_S, no adversary can create a fresh query packet without a successful MAC forgery. Packets can neither be replayed across sessions due to the randomly picked session identifier nor be replayed within a session due to the increasing sequence number.

5) Latency of Protocol: For the client-KS protocol, we implement FDH-RSA (RSA1024) using SHA256 in the standard way. Similar to [20], the PKI setup uses RSA2048 certificates and ECDHE-RSA-AES128-SHA ciphersuite is fixed for the handshake in our protocol. In our implementation, the client machine is located in a university LAN and equipped with Linux system (more precise, 2.6.35-22-generic version) with an Intel(R) Core(TM) 2 Duo CPU of 3.33 GHZ and 2.00-GB RAM. Table.I shows the latency of different phases in the form of median time plus/minus one standard deviation over 500 trials. We can note that when the KS load is low (e.g., 1000 queries/second (q/s)), the latency is quite small and actually almost the smallest possible time (1 Round-Trip Time (RTT) of Ping operation). The time increases with the growth of the query rate. Specifically, the latency is around 157±40 milliseconds when the query rate is 3000 q/s and becomes 376±44 milliseconds when the query rate increases to 7000 q/s. It is worth noting that we only take the successful operations into account in our experiment. That is, all the replies that timed out three times were excluded from the median calculation.

6) Performance Against On-Line KGA: To evaluate how our rate-limiting mechanism works in real settings, we estimate the effect of on-line KGA against our protocol in experiments. In our protocol, the proposed rate-limiting mechanism, i.e., exponential delay mechanism, gives a balancing between on-line KGA speed and KS request latency, as the delay increases exponentially with the growth of queries from a client. For the exponential delay mechanism in our experiment, we set the initial small delay t_I to 10 milliseconds and the epoch duration t_E to one week. We then evaluate the performance of protocol, i.e., the maximum query rates (in queries per second) by setting the upper limit t_U to different values. To evaluate the effectiveness of the introduced mechanism, we also run the protocol without rate-limiting. The maximum query rates that an attacker who compromised a client can achieve are given in Table.II. One can note from the result that our exponential delay mechanism can significantly slow down the on-line KGA. Specifically, the attack rate is around 2700 q/s if we put no rate-limiting on the KS. By forcing the exponential delay mechanism, the attack rate can be significantly reduced to less than 10 q/s. The attack rate decreases with the growth of the upper limit t_U and is only 1.21 q/s if we set t_U to 1000 ms.

TABLE II
KGA RATE FOR DIFFERENT RATE LIMITING APPROACHES

Setting	**Attack Rate** (queries/second)
No Rate Limiting	2700
$t_U = 400$ ms	8.23
$t_U = 600$ ms	4.21
$t_U = 800$ ms	2.54
$t_U = 1000$ ms	1.21

B. The Instantiated Scheme

1) Comparison of Schemes:

a) Computation: As shown in Table III, compared to the BCOP scheme [3] (the underlying PEKS scheme of our SA-PEKS construction), our scheme requires 4 additional RSA exponentiations during the generation of PEKS ciphertext and trapdoor. In the testing phase, our scheme has the same computation cost as the BCOP scheme. While the scheme [5] can also achieve a certain level of security against off-line KGA, its computation cost is much higher due to the additional pairing computation. Precisely, in our scheme, the computation cost of PEKS generation, trapdoor generation and testing are $2\mathrm{Exp}_{\mathbb{G}_1}$+$4\mathrm{Exp}_{\mathbb{Z}_N^*}$+$2\mathrm{Hash}_{\mathbb{G}_1}$+$1\mathrm{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$, $1\mathrm{Hash}_{\mathbb{G}_1}$+$1\mathrm{Exp}_{\mathbb{G}_1}$+$4\mathrm{Exp}_{\mathbb{Z}_N^*}$ and $1\mathrm{Hash}_{\mathbb{G}_1}$+$1\mathrm{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$ respectively, where $\mathrm{Exp}_{\mathbb{G}_1}, \mathrm{Exp}_{\mathbb{Z}_N^*}$ denote the computation of one exponentiation in $\mathbb{G}_1$ and $\mathbb{Z}_N^*$ respectively, $\mathrm{Hash}_{\mathbb{G}_1}$ denotes the cost of one hashing operation in $\mathbb{G}_1$.

b) Communication: We compare the communication cost of our scheme and the schemes in [3] and [5] in terms of the PEKS ciphertext, the trapdoor and the matching data set returned to the receiver (denoted as Ω in Table III). To be more precise, comparing to the underlying PEKS scheme [3], our scheme requires two additional $\mathbb{Z}_N^*$ elements transmitted between the KS and the user per generation of the PEKS ciphtertext or the trapdoor. We remark that this result is independent of the underlying PEKS scheme, as our solution works transparently with any existing PEKS system. Note that in our implementation of the client-KS protocol described above, we utilize FDH-RSA (RSA1024) and hence the corresponding communication overhead for each user is 2048 bits. The scheme presented in [5] requires two PEKS ciphertexts for each keyword and thus the size of PEKS ciphertext is doubled compared to the BCOP scheme [3]. Moreover, the data set transferred from the SS to the receiver is of the same size (i.e., Ω) for our scheme and the BCOP scheme while it is 2Ω or 3Ω for the scheme [5].

c) Storage: Regarding the storage cost, our scheme only introduces very small overhead for each user. That is, each user needs to store the public parameters (i.e., $N, \widehat{e}, H$) of the FDH-RSA blind signature to obtain the KS-derived keyword before the computation of each PEKS ciphertext and trapdoor. It is worth noting that the scheme [5] requires the receiver to

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARISONS BETWEEN EXISTING WORKS AND OUR SCHEME

Schemes	Computation			Communication		
	PEKS Ciphertext	*Trapdoor*	*Testing*	*PEKS Ciphertext*	*Trapdoor*	*Testing*
BCOP [3]	$2\text{Exp}_{\mathbb{G}_1}$ $+2\text{Hash}_{\mathbb{G}_1}$ $+\ 1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$1\text{Hash}_{\mathbb{G}_1}$ $+1\text{Exp}_{\mathbb{G}_1}$	$1\text{Hash}_{\mathbb{G}_1}+$ $1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$\mathbb{G}_1 + \log p$	$\mathbb{G}_1$	Ω
XJWW [5]	$4\text{Exp}_{\mathbb{G}_1}$ $+4\text{Hash}_{\mathbb{G}_1}$ $+2\ \text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$2\ \text{Hash}_{\mathbb{G}_1}$ $+2\text{Exp}_{\mathbb{G}_1}$	$2\text{Hash}_{\mathbb{G}_1}+$ $2\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$2\mathbb{G}_1 + 2\log p$	$\mathbb{G}_1$	2Ω or 3Ω
Our Scheme	$2\text{Exp}_{\mathbb{G}_1}$ $+2\text{Exp}_{\mathbb{Z}_N^*}$ $+2\text{Hash}_{\mathbb{G}_1}+\ 1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$1\text{Hash}_{\mathbb{G}_1}$ $+1\text{Exp}_{\mathbb{G}_1}$ $+2\text{Exp}_{\mathbb{Z}_N^*}$	$1\text{Hash}_{\mathbb{G}_1}+$ $1\text{Pairing}_{\mathbb{G}_1,\mathbb{G}_T}$	$2\mathbb{Z}_N^* + \mathbb{G}_1 + \log p$	$2\mathbb{Z}_N^* + \mathbb{G}_1$	Ω

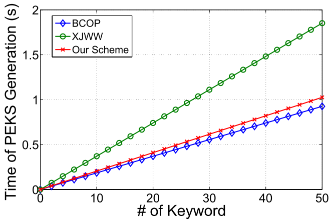


Fig. 4. Computation Cost of PEKS Generation (excluding latency).

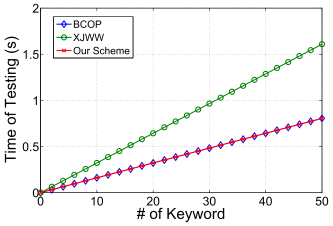


Fig. 6. Computation Cost of Testing.

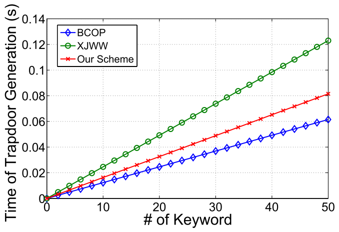


Fig. 5. Computation Cost of Trapdoor Generation (excluding latency).

keep the exact trapdoor per query in order to filter out the non-matching data from the set from the SS.

2) Experiment Results: To evaluate the efficiency of our scheme in experiments, we implement the scheme utilizing the GNU Multiple Precision Arithmetic (GMP) library and Pairing Based Cryptography (PBC) library. The following experiments are based on coding language C on a Linux system (more precise, 2.6.35-22-generic version) with an Intel(R) Core(TM) 2 Duo CPU of 3.33 GHZ and 2.00-GB RAM. For the elliptic curve, we choose an MNT curve with a base field of size 159 bits, $|p|$=160 and $|q|$=80.

We mainly analyze the computation cost of PEKS generation, trapdoor generation and testing in the schemes of [3] and [5] and our scheme. As shown in Fig.4 and Fig.5, the computation cost of our scheme is only slightly higher than that of the BCOP scheme in terms of PEKS generation and trapdoor generation. It is because that the computation involved in the underlying FDH-RSA scheme is quite small. Since our solution does not introduce any additional operation in the testing phase, the corresponding computation cost remains the same as the undeylying PEKS system, as illustrated in Fig. 6.

As for the scheme in [5] which achieves a certain level of security against off-line KGA, the computation cost is more than that of the PEKS scheme in [3] and our scheme in terms of all the operations. Particularly, it takes about 2 seconds to generate a PEKS ciphertext for the scheme in [5] when the keyword number is 50, while that of the scheme in [3] and our scheme is around 0.9 second and 1 second, respectively. For the trapdoor generation, the computation is slightly higher than that of our scheme as the exponentiation in $\mathbb{G}_1$ is usually more expensive than the exponentiation in $\mathbb{Z}_N^*$. To be more precise, the time of trapdoor generation for 50 keywords in [5] is about 0.12 seconds while that of our scheme is 0.08 seconds. Regarding the testing operation, the computation cost in [5] is almost twice that of our scheme. Specifically, the computation cost of testing is around 1.6 second for the scheme in [5] and 0.8 seconds for our scheme. This is because the testing in [5] requires an additional pairing computation.

VI. CONCLUDING REMARKS

In this work, we provided a practical and applicable treatment on (inside) off-line KGA by formalizing a new PEKS system, namely Server-Aided Public Key Encryption with Keyword Search (SA-PEKS). We introduced a universal transformation from any PEKS scheme to a secure SA-PEKS scheme, also with the first instantiation of SA-PEKS scheme and showed how to securely implement the client-KS protocol with a rate-limiting mechanism against on-line KGA. The experimental results showed that our proposed scheme achieves much better efficiency while providing resistance against both off-line and on-line KGAs.

ACKNOWLEDGEMENT

The authors thank the anonymous reviewers for their insightful feedbacks on this work.

REFERENCES

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May. 2000, pp. 44–55.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM CCS*, 2006, pp. 79–88.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.

[3] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, 2004, pp. 506–522.

[4] X. Yi, E. Bertino, J. Vaidya, and C. Xing, "Private searching on streaming data based on keyword frequency," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 2, pp. 155–167, Mar./Apr. 2014.

[5] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.

[6] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Proc. ACISP*, 2015, pp. 59–76.

[7] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2016.

[8] M. Abdalla *et al.*, "Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions," in *Proc. CRYPTO*, 2005, pp. 205–222.

[9] D. Khader, "Public key encryption with keyword search based on K-resilient IBE," in *Proc. Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2006, pp. 298–308.

[10] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, "Generating searchable public-key ciphertexts with hidden structures for fast keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1993–2006, Sep. 2015.

[11] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2008, pp. 1249–1259.

[12] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Improved searchable public key encryption with designated tester," in *Proc. ASIACCS*, 2009, pp. 376–379.

[13] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. 3rd VLDB Workshop Secure Data Manage. (SDM)*, 2006, pp. 75–83.

[14] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proc. ATC*, 2008, pp. 100–105.

[15] J. Baek, R. Safavi-Naini, and W. Susilo, "On the integration of public key data encryption and public key encryption with keyword search," in *Proc. 9th Int. Conf. Inf. Secur. (ISC)*, 2006, pp. 217–232.

[16] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electron. Exp.*, vol. 6, no. 5, pp. 237–243, 2009.

[17] D. Chaum, "Blind signature system," in *Proc. CRYPTO*, 1983, p. 153.

[18] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, 2010.

[19] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-rsa-inversion problems and the security of Chaum's blind signature scheme," *J. Cryptol.*, vol. 16, no. 3, pp. 185–215, 2003.

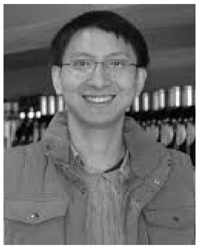
[20] S. Keelveedhi, M. Bellare, and T. Ristenpart, "DupLESS: Server-aided encryption for deduplicated storage," in *Proc. USENIX Secur.*, 2013, pp. 179–194.

Rongmao Chen received the B.S. and M.S. degrees in computer science from the National University of Defense Technology, China, in 2011 and 2013, respectively. He is currently pursuing the full-time Ph.D. degree with the School of Computing and Information Technology, University of Wollongong, Australia. His major research interests include cryptography, data security and privacy in cloud computing, and network security.

Yi Mu (SM'03) received the Ph.D. degree from the Australian National University in 1994. He is currently a Professor and the Co-Director of the Centre for Computer and Information Security Research with the University of Wollongong, Australia. His current research interests include information security and cryptography. He is also the Editor-in-Chief of the *International Journal of Applied Cryptography*. He serves as an Associate Editor for ten other international journals. He is a member of the IACR.

Guomin Yang received the Ph.D. degree in computer science from the City University of Hong Kong in 2009. He was a Research Scientist with the Temasek Laboratories, National University of Singapore from 2009 to 2012. He is currently a Senior Lecturer and a DECRA Fellow at the School of Computing and Information Technology, University of Wollongong. His research mainly focuses on applied cryptography and network security.

Fuchun Guo received the B.S. and M.S. degrees from Fujian Normal University, China, in 2005 and 2008, respectively, and the Ph.D. degree from the University of Wollongong, Australia, in 2013. He is currently an Associate Research Fellow with the School of Computing and Information Technology, University of Wollongong. His primary research interest is the public-key cryptography, in particular, protocols, encryption and signature schemes, and security proof.



Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently a Professor with the School of Mathematics and Computer Science, Fujian Normal University, China, and the Co-Director of Fujian Provincial Key Laboratory of Network Security and Cryptology. His research interests include applied cryptography and network security. He has authored over 100 research papers in refereed international conferences and journals. His work has been cited over 1900 times at Google Scholar (H-Index:25). He is an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING. He serves on the Editorial Board of *International Journal of Information Security* (Springer). He has served as the Program/General Chair or a Program Committee Member in over 60 international conferences.

Xiaofen Wang received the M.S. and Ph.D. degrees from Xidian University, Xian, China, in 2006 and 2009, respectively. She is currently a Lecturer with the School of Computer Science and Engineering and Big Data Research Center, University of Electronic Science and Technology of China, Chengdu, China. Her research interests are public key cryptography and its applications in wireless networks, smart grid, and cloud computing.

Yongjun Wang received the Ph.D. degree in computer architecture from the National University of Defense Technology, China, in 1998. He is currently a Professor with the College of Computer, National University of Defense Technology, Changsha, China. His research interests include network security and system security.

Authorized licensed use limited to: University of Wollongong. Downloaded on October 18,2022 at 09:24:51 UTC from IEEE Xplore. Restrictions apply.