Privacy-Preserving Image Denoising From External Cloud Databases

Yifeng Zheng, *Student Member, IEEE*, Helei Cui, *Student Member, IEEE*, Cong Wang, *Member, IEEE*, and Jiantao Zhou, *Member, IEEE*

***Abstract*— Along with the rapid advancement of digital image processing technology, image denoising remains a fundamental task, which aims to recover the original image from its noisy observation. With the explosive growth of images on the Internet, one recent trend is to seek high quality similar patches at cloud image databases and harness rich redundancy therein for promising denoising performance. Despite the well-understood benefits, such a cloud-based denoising paradigm would undesirably raise security and privacy issues, especially for privacy-sensitive image data sets. In this paper, we initiate the first endeavor toward privacy-preserving image denoising from external cloud databases. Our design enables the cloud hosting encrypted databases to provide secure query-based image denoising services. Considering that image denoising intrinsically demands high quality similar image patches, our design builds upon recent advancements on secure similarity search, Yao's garbled circuits, and image denoising operations, where each is used at a different phase of the design for the best performance. We formally analyze the security strengths. Extensive experiments over real-world data sets demonstrate that our design achieves the denoising quality close to the optimal performance in plaintext.**

***Index Terms*— Image denoising, external database, cloud computing, security, privacy.**

I. Introduction

THE rapid development of digital imaging technologies and the proliferation of various imaging devices have accelerated the explosive generation of images today. Along with the evolvement of many image processing tasks, image denoising, which refers to the procedure to recover the original image from its noisy observation, remains a fundamental one [1], [2]. Image noise may be caused by different intrinsic (i.e., sensor) and extrinsic (i.e., environment) conditions that are often hard to avoid in practical situations [3], [4]. Because obtaining the high quality original image content can be crucial in many contexts, image denoising serves as a stepping stone for a wide range of applications, such as image restoration, image registration, and image segmentation [3], [5].

Over the extensively studied literature, the class of patch-based image denoising algorithms is among the most highly-regarded ones, to date [1], [6]. Its idea is to find high quality similar patches and exploit them accordingly to perform patch-wise denoising of a noisy image [1], [7]. Traditionally, patch-based image denoising seeks similar patches from the noisy image itself, which sometimes has a quite limited search range subject to the redundancy within the image. The more popular trend nowadays is to harness similar patches from generic databases or targeted databases [1], [6]–[8], which are usually hosted at cloud for the appealing benefits which include but are limited to: relief of the burden for local storage management, broad network access, and avoidance of capital expenditure on hardware, software, and personnel maintenances [9], [10]. These appealing service benefits bring an emerging trend toward increasingly deploying various kinds of image services at cloud (e.g., [11]–[13], to just list a few). The external databases at cloud often provide much more information due to the large volumes of images, and thus have larger potential to yield more promising denoising performance [14], [15].

While the benefits of exploiting external databases are well understood, such a paradigm of cloud-based denoising would also raise security and privacy challenges. This is because many image datasets, e.g., medical images that contain diagnostic results, are inherently privacy-sensitive. When storing them at cloud, which is known to be facing wide attacking surfaces [16], [17], encryption would become an inevitable choice. Under such circumstances, how to enable privacy-preserving image denoising from the mandatorily encrypted cloud databases becomes of paramount importance. With such a privacy-preserving design, users should be able to use the query-based image denoising services at cloud while being worry-free. Ideally all data leaving from and arriving at the users' local device should always be encrypted.

In light of the above observations, in this paper we initiate the first study for privacy-preserving image denoising from

Manuscript received May 26, 2016; revised November 28, 2016; accepted January 9, 2017. Date of publication January 25, 2017; date of current version March 1, 2017. This work was supported in part by Research Grants Council of Hong Kong under Grant CityU 138513 and Grant CityU 11276816, in part by the Natural Science Foundation of China under Grant 61572412 and Grant 61402547, in part by the Innovation and Technology Commission of Hong Kong under ITF Project ITS/307/15, in part by the AWS Education Research Grant, in part by the Macau Science and Technology Development Fund under Grant FDCT/046/2014/A1, and in part by the Research Committee at the University of Macau under Grant MYRG2014-00031-FST, Grant MYRG2015-00056-FST, and Grant MYRG2016-00137-FST. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Shouhuai Xu.

Y. Zheng, H. Cui, and C. Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with the City University of Hong Kong Shenzhen Research Institute, Shenzhen 518057, China (e-mail: yifeng.zheng@my.cityu.edu.hk; helei.cui@my.cityu.edu.hk; congwang@cityu.edu.hk).

J. Zhou is with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau 999078, China (e-mail: jtzhou@umac.mo).

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.2017.2656824

external cloud databases. Considering that image denoising intrinsically demands high quality similar image patches, our design builds upon recent advancements on secure similarity search, Yao's garbled circuits, and image denoising operations. Specifically, we first consider the challenging question on searching high quality similar patches efficiently from the encrypted cloud database of image patches. At the first glance, such a task can be seemingly handled via the generic technique that combines searchable symmetric encryption (SSE) and locality-sensitive hashing (LSH) [18]–[20]. LSH is a well-studied algorithm which hashes and groups high-dimensional similar data records with high probability. By treating LSH values as keywords, it is possible to use SSE framework, which is originally for encrypted keyword search, to enable similarity search over the encrypted image patch database [18], [19].

However, LSH is an approximation algorithm that trades accuracy for efficiency, which usually locates a large number of candidates with false positives introduced. Thus, the search result of such a generic approach is often coarse-grained, and does not meet the stringent requirements for high quality similar patches in image denoising. Besides, the lack of ability to differentiate the quality of candidates of similar patches also invalidates any further meaningful encrypted denoising operations that would possibly follow up. These considerations drive us to further investigate how to securely filter the false-positive candidate patches and obtain similar patches accurately from the encrypted coarse-grained search result. Trivially sending all encrypted candidates to the users for local post-processing [18], [19], such as decryption, distance computation, and threshold-based evaluation, is not practically viable. One possible improvement is to let cloud assist the encrypted distance computation via homomorphic encryption [21]. However, our later analysis shows this is also expensive in practice, due to the burdensome user interactions with cloud as well as the high storage overhead at cloud.

In order to securely achieve high quality image patches while minimizing the local cost, we propose to build our services under the model of two non-colluding servers. We note that using the two-server model has been popular in many recent privacy-preserving designs which aim to achieve both strong security and practical performance. Some examples under such a model include ORAM [22], privacy-preserving matrix factorization [23], [24], secure deduplication [25], encrypted image feature extraction [26]–[28], and secure near-duplicate detection [29]. The two-server model allows us to exploit recent advancements on Yao's garbled circuits [30] to develop secure computation protocols and obtain high quality similar patches from encrypted candidates without interactions with the user. In particular, besides the cloud, the extra server we introduce is mainly responsible for generating our customized garbled circuit, which is to be securely evaluated by the cloud for obtaining high quality similar patches. Our implementation based on the advanced secure multi-party computation framework ObliVM [31] demonstrates the effectiveness and efficiency of the proposed design.

Furthermore, we investigate how to achieve effective denoising based on the high quality similar patches obtained. Typically, there are two types of operations for denoising based on similar patches, i.e., non-linear operations and linear operations [1]. Both can be directly supported by our design, because after secure filtering of the false-positive candidate patches, the user can conveniently retrieve all the high quality patches to perform denoising operations thereafter. Besides, for the linear operation based denoising, we also show how to optionally shift a large amount of local denoising workload to the cloud. Our extensive experiments demonstrate that the denoising quality achieved by our secure design is close to the optimal performance in the plaintext domain. Our contributions can be summarized as follows:

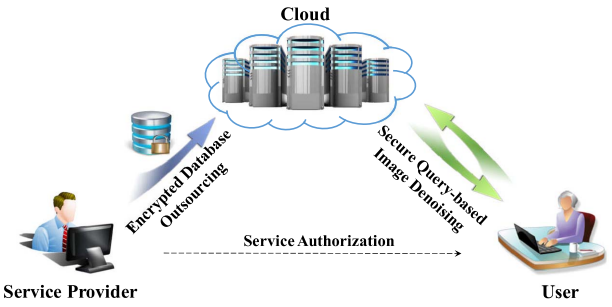


Fig. 1. The service model in our proposed design.

- To the best of our knowledge, we are the first that propose privacy-preserving image denoising from external cloud databases. Specifically, our proposed design enables the cloud hosting encrypted databases to provide secure query-based image denoising services.
- We design and implement a secure computation protocol based on Yao's garbled circuits to ensure that high quality similar patches are accurately obtained after encrypted similarity search, so as to achieve quality-ensured denoising.
- We formally analyze the security guarantees of our design and extensive experiments over real-world datasets demonstrate the effectiveness of our design. We show that our proposed design can achieve the denoising quality close to directly computing in the plaintext domain.

The rest of this paper is organized as follows. Section II presents our problem statement. Section III introduces some preliminaries. Section IV gives the details of our design. Section V presents the security analysis, followed by experiments in Section VI. Section VII describes the related work. Section VIII concludes the whole paper.

II. Problem Statement

A. Service Model

The basic service model targeted by our design is illustrated in Fig. 1. At the core, it contains three parties: the service provider (SP), the user, and the cloud. The SP outsources an encrypted database of image patches to the cloud, and wants to offer some secure "query" based image denoising service to authorized users. For example, online image sharing websites may construct a database with their collected images, and deploy its encrypted version at the public cloud to provide image denoising services for authorized users; a hospital can deploy an encrypted database at the cloud to assist the doctors

to denoise noisy medical images for improving diagnosis. The user can issue encrypted "queries" to the cloud, hoping to find high quality similar patches for image denoising and/or ask the cloud to further perform as many operations as possible over the encrypted search results. At certain points, the cloud may interact with an extra server which belongs to separate administrative domain, e.g., another independent cloud entity, to assist the completion of encrypted image denoising operations.

Note that although some public image datasets are available, they may not be necessarily suited to accommodate various kinds of image denoising applications and could be limited for practical use. Our proposed design allows the SP to either construct a generic database of patches extracted from a very comprehensive set of self-own images, thus with very rich structures, or customize a targeted database which is made from self-own images that are relevant to noisy images [1]. On another hand, even if public image datasets are deployed for denoising, it is of paramount importance to protect user's query images against the cloud. Therefore, an encrypted service design would still be desirable and necessary.

B. Threat Model

We consider the SP's image patch database and the user's query image patches to be private. Note that image patches could be used to reconstruct original private images, so they should be well protected [32], [33]. Our security goal is to ensure that neither the cloud nor the extra server learns such private data. We consider a semi-honest cloud, which honestly follows the protocol specification, yet is curious in inferring the private data. Besides, we assume that the semi-honest extra server does not collude with the cloud. That is, each of them is curious in inferring the private data, but they do so independently. The intuition behind the non-collusion assumption is that most cloud service providers are well-established and they are unlikely to collude with each other in order to maintain their own reputation and financial interests [26], [27], [34]. As mentioned before, such a non-colluding multi-server model has been commonly adopted in the literature for various security-aware applications.

We do not consider that the cloud compromises the integrity of the encrypted database and encrypted queries. In addition, the SP and the user are assumed to be trustworthy. The SP is responsible for user authorization, so that the user can directly generate legitimate queries. We assume that the authorization between the SP and the user is appropriately done.

III. Preliminaries

A. Image Denoising

Image denoising is a fundamental image processing problem, which serves as a stepping stone for various image-centric applications. The goal of image denoising is to restore a clean image from its noise-corrupted version. Mathematically, a noisy image $\mathbf{Y}$ is defined by

$$\mathbf{Y} = \mathbf{X} + \mathbf{E}, \tag{1}$$

where $\mathbf{X}$ is the original clean image, and $\mathbf{E}$ is the additive white Gaussion noise (AWGN) with standard deviation σ [35]. Therefore, given a noisy image $\mathbf{Y}$, image denoising aims to estimate the original image $\mathbf{X}$ as accurately as possible.

Numerous image denoising algorithms have been designed during the past decades, among which the class of patch-based image denoising algorithms is highly regarded [1]. Given a patch $\mathbf{q}$ of a noisy image, a patch-based denoising algorithm finds a set of similar patches $S = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k\}$, and then applies some linear or non-linear operation ψ to produce an estimate $\widehat{\mathbf{p}}$ of the unknown clean patch $\mathbf{p}$, i.e., $\widehat{\mathbf{p}} = \psi(\mathbf{q}; S)$. By iterating the same process for each patch of the noisy image, an estimate of the original image can be produced.

According to the source of similar patches, patch-based denoising algorithms can be generally classified into two types: internal denoising and external denoising [1]. Internal denoising searches similar patches within the noisy image, while external denoising searches similar patches from an external database. External denoising is a popular trend [1], [6], [14], [15], which, in theory, has been proved to be able to achieve the minimum mean squared estimation error [14], [15], i.e., the optimal denoising quality. In this paper, we focus on external denoising and investigate how to enable privacy-preserving image denoising from encrypted cloud databases.

Similar to prior work [36], we adopt the squared Euclidean distance to measure the patch similarity, which is defined by

$$d(\mathbf{q}, \mathbf{p}_i) = \|\mathbf{q} - \mathbf{p}_i\|_2^2, \tag{2}$$

where $\|\cdot\|_2^2$ denotes the squared Euclidean distance. Given a patch $\mathbf{q}$, similar patches are defined as those ones whose distances from $\mathbf{q}$ are within a pre-defined threshold.

Collecting the similar patches, we will focus on the linear operation as the first instantiation. Specifically, we adopt the operation of weighted average as in one of the most influential denoising techniques called non-local means (NLM) denoising [6], [37]. Specifically, given a noisy patch $\mathbf{q}$ and a set of similar patches $S = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k\}$, the clean patch $\mathbf{p}$ is estimated as the weighted average of all similar patches, i.e.,

$$\widehat{\mathbf{p}} = \sum_{i=1}^{k} w(\mathbf{q}, \mathbf{p}_i) \cdot \mathbf{p}_i. \tag{3}$$

The weight $w(\mathbf{q}, \mathbf{p}_i)$ is calculated as

$$w(\mathbf{q}, \mathbf{p}_i) = \frac{1}{Z} e^{-\frac{\|\mathbf{q}-\mathbf{p}_i\|_2^2}{h^2}}, \tag{4}$$

where Z is the normalizing factor such that

$$Z = \sum_{i=1}^{k} e^{-\frac{\|\mathbf{q}-\mathbf{p}_i\|_2^2}{h^2}}, \tag{5}$$

and h is a filtering parameter depending on the standard deviation σ of the zero-mean Gaussian noise. Note that σ can be estimated from the noisy image through various effective methods [38]–[40] and thus in the literature, σ is treated as a known prior, serving as an input to a denoising algorithm [1], [35], [36]. Therefore, we also consider σ as a known priori.

B. Yao's Garbled Circuits

Yao's garbled circuits enables two parties holding inputs a and b, respectively, to jointly compute an arbitrary function $\mathsf{fun}(a, b)$ without leaking any information about their inputs beyond what is implied by the function output [30], [41]. Specifically, one party called the generator first prepares an garbled version of a circuit which computes $\mathsf{fun}(\cdot, \cdot)$. Then, the generator provides the garbled circuit and the garbled input $\widehat{a}$ corresponding to a for the other party called the evaluator. The evaluator runs a 1-out-of-2 oblivious transfer (OT_1^2) protocol with the generator to obliviously obtain the garbled input $\widehat{b}$ corresponding to its private input b. From $\widehat{a}$ and $\widehat{b}$, the evaluator can evaluate the garbled circuit to obtain the result of $\mathsf{fun}(a, b)$.

IV. Privacy-Preserving Image External Denoising

This section presents our design of privacy-preserving image denoising from external cloud databases. Before giving our main result, we first study some closely related existing results from encrypted similarity search and encrypted distance calculation. Through detailed analysis, we show that these basic approaches, while serving as good starting points, are not able to address our problem completely. The analysis of their demerits will lead to our main result, which strikes a balance between security and practicality.

A. The Basic Approaches

For encrypted similarity search, existing works have shown that it can be achieved via the direct combination of SSE [19], [42], [43] and LSH [18], [19]. Building upon these works, we can let the cloud do the similarity search over the encrypted image patch database. The good efficiency of such a design serves as a good starting point for us. However, as said previously, the search result of such a generic approach often includes false-positive candidates, which are usually unavoidable due to the approximation nature of LSH. Consequently, for the betterment of image denoising quality, a step of filtering false positives has to be enforced. Roughly speaking, there are two possible ways to achieve this.

1) First Attempt: The first trivial one is to let the cloud directly return the candidate patch ciphertexts, then the user decrypts them, computes the distances, and further evaluates which candidate patches satisfy the distance metric. In this solution, the returned patch ciphertexts only need to be protected under symmetric encryption (SE) (e.g., AES), yet the transmission of a large number of patch ciphertexts could incur cumbersome bandwidth cost and undesirable post-processing.

2) Second Attempt: Instead of returning encrypted patches for user-side filtering, the second possible way is to let the cloud return encrypted distances to the user, who then performs decryption and distance evaluation. This requires encrypted distance calculation at the cloud. Instead of using costly full homomorphic encryption, we observe that the approach proposed by a recent work [21] can be adopted, which is based on additively homomorphic encryption (AHE) (e.g., the Paillier cryptosystem).

It works as follows. Given a database patch $\mathbf{p}_i=\{p_{i,j}\}_{j=1}^n$,[1] the SP generates the AHE ciphertexts $\{[p_{i,j}^2]\}_{j=1}^n$ and $\{[-r_j \cdot p_{i,j}]\}_{j=1}^n$, where $\{r_j\}_{j=1}^n$ is a randomly generated secret vector. To submit a query for a noisy patch $\mathbf{q}=\{q_j\}_{j=1}^n$, in addition to the search token, the user produces the AHE ciphertexts $\{[q_j^2]\}_{j=1}^n$ and $\{r_j^{-1} \cdot q_j\}_{j=1}^n$. To calculate the encrypted distance $[d(\mathbf{q}, \mathbf{p}_i)]$, for $j \in [1, n]$, the cloud first computes $[-r_j \cdot p_{i,j}]^{2r_j^{-1} \cdot q_j}=[-2q_j \cdot p_{i,j}]$ and $[q_j^2][p_{i,j}^2][-2q_j \cdot p_{i,j}]=[(q_j - p_{i,j})^2]$. Then, the cloud calculates $[d(\mathbf{q}, \mathbf{p}_i)]=\prod_{j=1}^n [(q_j - p_{i,j})^2]$.

After calculating the encrypted distances between the query patch and candidate patches, the cloud can send them to the user, who then performs decryption and distance evaluation. However, it has the following demerits: 1) the storage consumption at the cloud side to support encrypted distance calculation could be cumbersome, as multiple AHE ciphertexts have to be stored for each database patch $\mathbf{p}_i$, i.e., $\{[p_{i,j}^2]\}_{j=1}^n$ and $\{[-r_j \cdot p_{i,j}]\}_{j=1}^n$. Suppose that the number of database patches is N. The storage cost for these ciphertexts is $2 \times a \times n \times N$, where a denotes the size of a Paillier ciphertext and typically $a=256$ bytes [44]. Such storage cost could be significant as the number of database patches grows large, which exactly is the trend in the era of big data. For example, for a very large N say 10^{10} [6] and $n=9 \times 9=81$, it would consume more than 370 TB storage, which could be cumbersome; 2) the transmission of a large number of AHE-protected distances may incur heavy bandwidth cost. Note that generally the size of AHE ciphertext is much larger than that of SE ciphertext. Returning the AHE-protected distances actually consumes more bandwidth than returning the SE-protected patch ciphertexts. In particular, the size of an AHE ciphertext is 256 bytes, which could be much larger than the patch ciphertext protected by SE like AES. For example, by AES-128 in CTR mode, the ciphertext for a 9×9 patch with pixels of 8-bit depth would be roughly 81 bytes.

B. Our Proposed Scheme

The above basic approaches demonstrates that user-guided filtering is not practically viable. A more desirable approach is to securely transfer the user-guided filtering to cloud-side filtering. That is, we want to have a design to enable the cloud to evaluate each candidate patch by itself after encrypted similarity search, while being free of interactions with the user.

In the literature, we observe that the non-colluding two-server model has been popular and is widely adopted to facilitate various secure applications, such as ridge regression [45], matrix factorization [23], [24], and image feature extraction [27]. Inspired by these works, we leverage the non-colluding two-server model to securely change user-guided filtering into cloud-side filtering, after we have obtained the large number of candidates from encrypted similarity search.

1) Design Overview: To filter false-positive candidates at the cloud side, we resort to the approach of Yao's garbled circuits, of which the performance has been steadily boosted over the years [31]. Specifically, the extra server we introduce

[1] For ease of exposition, each patch is represented as a vector of n elements.

prepares garbled circuits for the cloud, who then acts as an evaluator to securely evaluate whether the distances between the query patch and candidate patches are within the threshold. With a secure garbled circuit based design that protects the patches against both the cloud and the extra server, we can enable the cloud to find out the similar patches for denoising securely and accurately, without interactions with the user.

a) Leveraging Yao's garbled circuits: The basic way to instantiate the garbled circuit based design for secure patch evaluation is as follows. First, the extra server produces an asymmetric key pair (pk_G, sk_G) of a public-key encryption scheme. Then, the public key pk_G is used by the SP to encrypt the database patches. When a user wants to query the encrypted database for image denoising, she provides the cloud with patches encrypted under pk_G, along with the search token. After encrypted similarity search, for each candidate patch, the extra server prepares a garbled circuit and deploys it to the cloud. The garbled circuit takes as input the garbled values corresponding to the encrypted candidate patch $[\mathbf{p}_{id}]$, query patch $[\mathbf{q}]$, and a threshold ϵ. The evaluation inside the circuit first decrypts the patch ciphertexts via the secret key sk_G, computes the distance $d(\mathbf{q}, \mathbf{p}_{id})$, and finally compares it with the threshold. This is a viable approach for secure evaluation at the cloud side, yet we note that it lacks good efficiency as decryption inside the circuit needs to be conducted [45].

b) Avoiding patch decryption within circuit: To avoid decryption inside the circuit, we use the random masking technique, inspired by [23], [29], [45]. The main idea is to let the cloud add random masks to the encrypted patches in the ciphertext domain, which can be realized if the patches are encrypted with AHE. In this way, the extra server only obtains masked patches after decrypting patch ciphertexts, and can prepare a garbled circuit inside which the decryption module is replaced by a lightweight subtraction module that removes the mask. Thus, the efficiency of secure evaluation can be improved.

c) Speedup via LSH-voting: Although the proper use of garbled circuits allows the cloud-side filtering of false-positive candidates, it does not directly enable a practical design and thus some optimization is required. Specifically, the number of candidate patches located via encrypted similarity search usually could be considerably large. Thus, it is not practical for the cloud to directly evaluate each candidate patch so as to find sufficient, say k, similar patches for the denoising of a query patch. Note that if we apply the Yao's garbled circuits design directly, the large number of possible candidates could directly trigger the same amount of interactions between the two servers, seriously downgrading the overall performance.

To address this issue, we further introduce a LSH-voting mechanism in our design, exploiting the fact that the more common LSH values two patches share, the more similar they are [46]. Specifically, our design applies multiple LSH functions, and uses a counter to record the occurrence of common LSH values that the query patch and each candidate patch share. Then, the candidate patches are ranked based on the counters. Based on the ranking result, secure patch evaluation now can be performed for the candidates in order. As long as k accurately similar patches are identified, the cloud doesn't need to evaluate the remaining candidates. With this practical optimization, the efficiency of the overall system performance can be significantly improved.

Algorithm 1 Building the Encrypted Database for Outsourcing

Input: Secret key: $\mathbf{K} = \{K_g, K_p\}$; Public key: pk_G; Patch set: $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$, where N is the total number of database patches.

Output: Encrypted database $\mathcal{E} = \{[\mathbf{P}], [\![\mathbf{P}]\!], \mathcal{D}\}$.

1: Initialize a generic dictionary $\mathcal{D}$ and the LSH value set $\mathbf{G}$;
2: **for all** $\mathbf{p} \in \mathbf{P}$ **do**
3: // Compute LSH values:
4: $\mathbf{g} = \{h_1(\mathbf{p})||1, \cdots, h_l(\mathbf{p})||l\}$, where $g_i = h_i(\mathbf{p})||i$;
5: $\mathbf{G}.put(\mathbf{g})$;
6: **end for**
7: **for all** $g \in \mathbf{G}$ **do**
8: $K_1 \leftarrow F(K_g, 1||g)$, $K_2 \leftarrow F(K_g, 2||g)$;
9: Initialize counter $ctr \leftarrow 0$;
10: **for all** $\mathbf{p}$ associated with g **do**
11: $u \leftarrow F(K_1, ctr)$;
12: $v \leftarrow \mathsf{SE.E}(K_2, id)$, where id is the unique identifier of a database patch;
13: $ctr++$;
14: $\mathcal{D}.insert(u, v)$;
15: **end for**
16: **end for**
17: $[\mathbf{P}] \leftarrow \mathsf{AHE.E}(pk_G, \mathbf{P})$, $[\![\mathbf{P}]\!] \leftarrow \mathsf{SE.E}(K_p, \mathbf{P})$;

2) Scheme Details: We now present the details of our proposed scheme which enables the user to securely obtain high quality similar patches from the encrypted cloud database. It supports the filtering of false-positive candidate patches at the cloud side. In particular, we will show the details of building the encrypted database, searching the encrypted database, and evaluating the candidate patches. Let $\mathsf{SE}=(\mathsf{SE.E}, \mathsf{SE.D})$ be the pair of encryption/decryption algorithms in an SE scheme, $\mathsf{AHE}=(\mathsf{AHE.E}, \mathsf{AHE.D})$ in an AHE scheme, and F be a pseudorandom function (PRF). The details of each scheme phase is as follows.

a) Building the encrypted database: To outsource an encrypted database that supports secure image denoising, the SP encrypts the patches and builds a secure index from the combination of LSH and SSE [43]. Multiple LSH functions are applied, i.e., $h_1(\cdot), h_2(\cdot), \cdots, h_l(\cdot)$, for the LSH-voting mechanism as described above. Algorithm 1 illustrates the secure index building process. Specifically, the SP operates as follows:

1) For each patch, generate the LSH values. Each of them is associated with a patch identifier id, which is used to not only locate the candidate patches, but also work for the LSH-voting mechanism;
2) For each LSH value, compute a pseudorandom tag u for each associated patch by applying a PRF to a counter ctr, and encrypt the corresponding patch identifier id. Then, insert the tag-ciphertext pair to a generic dictionary $\mathcal{D}$;
3) Output the dictionary $\mathcal{D}$ as the secure index.

For patch encryption, the SP produces two ciphertexts $[\![\mathbf{p}]\!]$=SE.E$(K_p, \mathbf{p})$ and $[\mathbf{p}]$=AHE.E$(pk_G, \mathbf{p})$ for each database patch $\mathbf{p}$, where K_p is a private key for patch encryption, and pk_G is the public key of the extra server. Note that as all AHE patch ciphertexts are protected under the same public key from the extra server, we might be able to assume that in practice, different data owners who want to contribute to the external cloud database can send both the plaintext patch and its corresponding AHE ciphertext to the SP. In this way, the computation cost of the SP can be alleviated.

b) Searching the encrypted database: The procedure of search over the encrypted patch database is illustrated in Algorithm 2. For each noisy patch $\mathbf{q}$, the user sends the cloud a secure search token, and an AHE ciphertext $[\mathbf{q}]$ which would get involved in secure patch evaluation. In particular, the protocol of encrypted patch search operates as follows:

1) To generate a secure search token for a query patch $\mathbf{q}$, the user first hashes $\mathbf{q}$ into a vector of LSH values. Then, for each LSH value, a value pair $\{K_1, K_2\}$ is generated via a PRF and the resulting secure search token Q consists of l such sub-tokens;
2) For each sub-token $\{K_1, K_2\}$, the cloud re-computes the pseudorandom tag via $F(K_1, ctr)$ and searches the generic dictionary $\mathcal{D}$ to locate the corresponding encrypted patch identifiers, which are then decrypted via K_2. Here, ctr is a self-incremental counter;
3) For each recovered patch identifier id, the cloud uses an occurrence counter f_{id} to record the number of common LSH values it shares with the query patch. Then, the cloud ranks the candidates based on the occurrence counters, and derives a initial set S^* of candidate patches.

c) Evaluating the candidate patches: To filter the false-positive candidates and identify k similar patches which satisfy the distance metric, the cloud then runs a protocol based on Yao's garbled circuits with the extra server, to securely evaluate each candidate according to its ranking in S^*. The secure patch evaluation protocol operates as follows:

1) For each encrypted candidate patch $[\mathbf{p}_{id}]$, the cloud chooses a random mask $\mathbf{r}_{id}$, and produces $[\mathbf{p}'_{id}] = [\mathbf{p}_{id}][\mathbf{r}_{id}] = [\mathbf{p}_{id}+\mathbf{r}_{id}]$. Similarly, for the encrypted query patch $[\mathbf{q}]$, the cloud produces $[\mathbf{q}'] = [\mathbf{q} + \mathbf{r}_q] = [\mathbf{q}][\mathbf{r}_q]$. Then the cloud sends $[\mathbf{q}']$ and $[\mathbf{p}'_{id}]$ to the extra server;
2) Upon receiving the processed patch ciphertexts, the extra server first uses its private key sk_G to decrypt them and derive the masked patches, i.e., $\mathbf{q}'=(\mathbf{q} + \mathbf{r}_q)$ and $\mathbf{p}'_{id}=(\mathbf{p}_{id} + \mathbf{r}_{id})$;
3) The extra server builds a garbled circuit consisting of three modules, i.e., mask subtraction, distance calculation, and threshold comparison. Besides, it generates the garbled input values for $\mathbf{q}'$ and $\mathbf{p}'_{id}$, i.e., $\widehat{\mathbf{q}'}$ and $\widehat{\mathbf{p}'_{id}}$. Then the extra server sends the garbled circuit along with the garbled input values $\widehat{\mathbf{q}'}$ and $\widehat{\mathbf{p}'_{id}}$ to the cloud;
4) The cloud runs an oblivious transfer protocol with the extra server to obtain the garbled values of the random masks $\mathbf{r}_q$, $\mathbf{r}_{id}$, and the threshold ϵ, i.e., $\widehat{\mathbf{r}_q}$, $\widehat{\mathbf{r}_{id}}$ and $\widehat{\epsilon}$;
5) The cloud evaluates the garbled circuit and derives a Boolean result, which indicates whether the candidate patch satisfies the distance metric or not.

Algorithm 2 Searching the Encrypted Database for Denoising

Input: Secret key: $\mathbf{K} = \{K_g, K_p\}$; Public key: pk_G; Query patch: $\mathbf{q}$.
Output: Candidate patch set S^*.

User: // Secure query generation
$\mathbf{g} = \{h_1(\mathbf{q})||1, \cdots, h_l(\mathbf{q})||l\}$;
for all $g \in \mathbf{g}$ **do**
  $K_1 \leftarrow F(K_g, 1||g)$, $K_2 \leftarrow F(K_g, 2||g)$;
end for
$[\mathbf{q}] \leftarrow$ AHE.E$(pk_G, \mathbf{q})$;
Send $\mathbf{Q} = \{K_1, K_2\}_l$ and $[\mathbf{q}]$ to the cloud;
Cloud: // Encrypted patch search
for all id **do**
  Initialize occurrence counter $f_{id} = 0$;
end for
for each $\{K_1, K_2\} \in \mathbf{Q}$ **do**
  Counter $ctr \leftarrow 0$;
  $v \leftarrow \mathcal{D}.find(F(K_1, ctr))$;
  while $v \neq NULL$ **do**
    $id \leftarrow$ SE.D(K_2, v);
    $f_{id}++$; $ctr++$;
    $v \leftarrow \mathcal{D}.find(F(K_1, ctr))$;
  end while
end for
Rank located patches based on f_{id} and output a ranked candidate patch set S^*

Note that the cloud terminates the process of secure patch evaluation when it has located k accurately similar patches. Besides, as the threshold ϵ is independent of the original image signal, it is no necessary to explicitly protect it against the cloud here. Yet, our design can also secure it by simply asking the user to encrypt it with pk_G before sending it to the cloud.

As false-positive candidates are filtered at the cloud side, the user can let the cloud return the accurately similar patches protected by SE and do all the other post-processing for denoising. In particular, the user can perform denoising through either linear or non-linear operation based denoising techniques. In our experiments, we have chosen the classical NLM technique as an exemplary instantiation on linear operation based denoising in our design. But our design readily allows the user to choose more sophisticated techniques for denoising as well, such as dictionary learning [35].

d) Discussion: We next show a possible design for the case of linear operation based denoising, which can further shift the local denoising workload from the user to the cloud at the price of weight information exposure. This approach is inspired by the second attempt with the help of additively homomorphic encryption. It goes as follows. First, the SP additionally stores an AHE ciphertext for each database patch $\mathbf{p}_i$, i.e., $[\mathbf{p}_i]$=$\{[\mathbf{p}_{i,j}]\}_{j=1}^{n}$. After retrieving the accurately similar

patches, the user computes the weight for each similar patch as defined in Eq. 4, and sends them to the cloud. Without loss of generality, we denote with $\{w_i\}_{i=1}^{k}$ the weights. With these weights, the cloud can now produce the encrypted denoised patch for the user by computing

$$[\hat{\mathbf{p}}] = \left[\sum_{i=1}^{k} \overline{w}_i \mathbf{p}_i\right] = \prod_{i=1}^{k} [\mathbf{p}_i]^{\overline{w}_i}, \tag{6}$$

where $\overline{w}_i = \gamma \cdot \lceil w_i \rfloor$ is the integer representation of w_i (γ is a scaling factor and $\lceil \cdot \rfloor$ is the rounding function), because AHE operates over integers. In this way, the user does not need to perform weighted average by herself.

We must note that the usage case of such kind of denoising is restricted to weighted average based applications. The saving on the local computation is actually traded off by the exposure of weight information as well as the extra size of AHE ciphertexts at cloud. Therefore, we remark that returning accurately similar SE-protected patches is often more flexible and can embrace various patch-based denoising techniques.

V. Security Analysis

In this section, we provide formal security analysis to demonstrate the security guarantees of our proposed scheme. Recall that the patches in the service flow of our design are protected by encryption schemes which are semantically secure. Besides, our design of secure patch evaluation based on Yao's garbled circuits and random masking ensures the security for the input patches against both the cloud and the extra server. And the output of secure patch evaluation only indicates whether the distance between a candidate patch and the query patch is within a threshold. Note that in our design, we do not consider that the semi-honest cloud might misbehave to evaluate the distances of the encrypted database patches, rather than the designated distances between the candidate patches and the query patch. Although such misbehavior might be addressed by commitment-based verification primitives as proposed in [23], [45], we leave it as our future work.

We mainly focus on analyzing the security of the interaction between the user and the cloud during encrypted patch search. In particular, we will follow the security framework of SSE [42], [43], [47]. Firstly, we formally characterize the leakages revealed by the encrypted database and from the interactions. Then, we follow the simulation-based paradigm to prove that our design is secure against adaptive chosen-keyword attacks in the random oracle model.

We want to point out that the security framework adopted by existing SSE schemes [42], [43], [47] allows access pattern leakage and search pattern leakage. Here, access pattern refers to the search result and the accessed content of the secure index, while search pattern indicates whether the query keyword has been used before. In addition, similar to existing works on encrypted similarity search [18], [48] bridging SSE and LSH, our design also allows the similarity leakages between the queries. Recall that the search token of one query contains multiple LSH values, and thus the number of common LSH values two queries share indicates their similarity. Formally, the leakage functions in our design are defined as follows:

1) The leakage $\mathcal{L}_{edb}$ from the encrypted database is

$$\mathcal{L}_{edb} = (M, |u|, |v|, |[\mathbf{p}]|/|\left[\left[\mathbf{p}\right]\right]|),$$

where M is the number of entries in the secure index $\mathcal{D}$, $|u|$ is the bit length of each tag in $\mathcal{D}$, $|v|$ is the bit length of each encrypted value corresponding to $|u|$, and $|[\mathbf{p}]|$ and $|\left[\left[\mathbf{p}\right]\right]|$ are the bit lengths of the encrypted patches $[\mathbf{p}]$ and $\left[\left[\mathbf{p}\right]\right]$, respectively.

2) The leakage $\mathcal{L}_{eps}$ from encrypted patch search is

$$\mathcal{L}_{eps} = (\{[\mathbf{p}_{id}]/\left[\left[\mathbf{p}_{id}\right]\right]\}_{id\in S^*}, \mathbf{N}_{s\times s}),$$

where $\{[\mathbf{p}_{id}], \left[\left[\mathbf{p}_{id}\right]\right]\}_{id\in S^*}$ is the access pattern for a given query, i.e., the search result of encrypted candidate patches; $\mathbf{N}_{s\times s}$ is a binary symmetric binary matrix that records the repeated queries. That is, the entry of $\mathbf{N}$ in the ith row and jth column is set to 1 if the ith query and the jth query are the same, and 0 otherwise.

Given the above characterized leakages, we now follow the security framework of [42], [43], [47] and give the simulation-based security definition as follows.

Definition 1: Let $\Pi = (\mathsf{KeyGen}, \mathsf{IndBuild}, \mathsf{PatchSrh})$ *be the secure index-based scheme of encrypted patch search,* λ *be the security parameter,* $\mathcal{A}$ *be an adversary, and* $\mathcal{S}$ *be a simulator. We define the following probabilistic experiments:*
$\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$: *The SP executes* KeyGen *to generate the private keys.* $\mathcal{A}$ *selects a set of image patches, and asks the SP to generate the index and the ciphertexts via the algorithm* IndBuild. *Then* $\mathcal{A}$ *launches a polynomial number of s queries, and asks the authorized user for the secure search tokens and the resulting patch ciphertexts via the algorithm* PatchSrh. *Finally,* $\mathcal{A}$ *outputs a bit.*
$\mathbf{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$: $\mathcal{A}$ *selects a set of image patches. Then based on the leakage* $\mathcal{L}_{edb}$, $\mathcal{S}$ *simulates an index and patch ciphertexts for* $\mathcal{A}$. $\mathcal{A}$ *launches a polynomial number of adaptive s queries. Based on the leakage* $\mathcal{L}_{eps}$, $\mathcal{S}$ *returns simulated search tokens and patch ciphertexts. Finally,* $\mathcal{A}$ *outputs a bit.*

We say that Π *is* $(\mathcal{L}_{edb}, \mathcal{L}_{eps})$*-secure against adaptive chosen keyword attacks if for all polynomial-time adversaries* $\mathcal{A}$*, there exists a simulator* $\mathcal{S}$ *such that* $|\Pr[\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq negl(\lambda)$*, where* $negl(\lambda)$ *is a negligible function in* λ.

We now prove that our design for encrypted patch search is secure against adaptive chosen keywords attacks with respect to the characterized leakages.

Theorem 1: Our design Π *for encrypted patch search is* $(\mathcal{L}_{edb}, \mathcal{L}_{eps})$*-secure against adaptive chosen-keyword attacks in the random oracle model if (*SE.E, SE.D*) and (*AHE.E, AHE.D*) are semantically secure, and F is a secure PRF.*

Proof: We prove the existence of a simulator $\mathcal{S}$ such that for all polynomial-time adversaries $\mathcal{A}$, the output of $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ are computationally indistinguishable. In particular, the simulator $\mathcal{S}$ can adaptively simulate an encrypted database $\widetilde{\mathcal{E}}$ (i.e., the secure index and patch ciphertexts), and s adaptive queries $(\widetilde{Q_1}, \widetilde{Q_2}, \ldots, \widetilde{Q_s})$ as follows:

1) *Simulating the index* $\mathcal{D}$: Given $\mathcal{L}_{edb}$, $\mathcal{S}$ initializes a dictionary $\widetilde{\mathcal{D}}$ with M entries. For each entry i, it inserts a random bit string $\widetilde{v}$ of size equal to $|v|$, which also corresponds to a random unique search tag $\widetilde{u}$ of size equal to $|u|$.
2) *Simulating the patch ciphertexts*: $[\![\mathbf{p}]\!]$ / $[\![[\![\mathbf{p}]\!]]\!]$: Given $\mathcal{L}_{edb}$, for each patch ciphertext, $\mathcal{S}$ simulates $\widetilde{[\![\mathbf{p}]\!]}/\widetilde{[\![[\![\mathbf{p}]\!]]\!]}$, which is a random bit string of size equal to $\big|[\![\mathbf{p}]\!]\big| / \big|[\![[\![\mathbf{p}]\!]]\!]\big|$.
3) *Simulating the query* Q: Given $\mathcal{L}_{eps}$, $\mathcal{S}$ can simulate the first query and its corresponding results. In particular, for each sub-token in the query Q_1, it generates a random string set $\{\widetilde{K_1}, \widetilde{K_2}\}$ as a simulated sub-token in the simulated query $\widetilde{Q}_1$, where the length of each one is the same as the real one. Then, $\mathcal{S}$ programs a random oracle to point at randomly selected tag-ciphertext pairs in $\widetilde{\mathcal{D}}$ and reveal the same identifiers to match the real ones observed from the leakage $\mathcal{L}_{eps}$. The identical number of simulated patch ciphertexts randomly assigned from $\widetilde{[\![\mathbf{p}]\!]}/\widetilde{[\![[\![\mathbf{p}]\!]]\!]}$ are considered as the simulated query results. The simulation can be extended to a number of adaptive queries. To simulate the subsequent queries, if $\mathcal{L}_{eps}$ indicates that the query is a repeated one, $\mathcal{S}$ will select the same entries in $\widetilde{\mathcal{D}}$ and use the same simulated tokens and results generated before. Otherwise, $\mathcal{S}$ will perform the same procedure as in simulating the first query.

The semantic security of symmetric encryption and homomorphic encryption, and the pseudorandomness of F ensures that the real encrypted database (i.e., index and patch ciphertexts) and the simulated one, and the real search tokens and the simulated ones are computationally indistinguishable. We want to point out that the simulated encrypted database has the identical structure with the real encrypted database. That is, the sizes of all data structures are the same, and random strings with equal length are filled in the structures. Besides, the search tokens in the simulated queries and the real ones have the same structure, and the simulated results and the real ones are identical. Therefore, the adversary $\mathcal{A}$ should not be able to distinguish between the real interactions and the simulated ones. The outputs of $\mathbf{Real}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Sim}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ are computationally indistinguishable. □

VI. Experiments

A. Implementations

We implement a preliminary system prototype in *Java*, and test it on AWS "c4.4xlarge" instances in Linux (Ubuntu Server 14.04.2 LTS). For image processing, we use the OpenCV library[2] to implement the functionalities of our patch extraction and image reconstruction. For patch encryption, we use the *Java Cryptography Architecture* (JCA) to realize the standard symmetric encryption via AES-128 in CTR mode, the PRF via HMAC-SHA1, and the Paillier library[3] for additively homomorphic encryption. Note that we pack multiple pixels into one Paillier ciphertext, so as to reduce the space overhead of ciphertexts [49]. Regarding the PRF, it is realized via the cryptographic hash function HMAC-SHA1 in the JCA.

To prove the concept of our garbled circuit based design, we use ObliVM-lang [31], the advanced framework for secure multi-party computation, to compile our customized algorithm and test the performance. We use E2LSH in our experiments, and the LSH parameter l is set to 10 by default for demonstration purpose, unless otherwise stated. We are aware that this parameter setting either has been adopted in some existing works (e.g., [50]), or has the same order of magnitude as some existing works (e.g., [18], [51]). Hence, we consider that the default setting $l = 10$ should be reasonable. Tuning optimal LSH parameters is not the focus of this paper and it could be addressed by some orthogonal works (e.g., [52], [53]).

In our experiments, following most prior works which use external database for image denoising [6], [8], [15], we will evaluate denoising by generic database. Besides, for completeness, we will also evaluate denoising by targeted database [1]. In particular, for denoising by targeted database, we denoise noisy face images by using patches extracted from face images. In the case of generic database, following [8], we use the Berkeley dataset[4] which has 300 generic images. We randomly select 200 images out of them, of which 100 are used as the test images, and the rest are used to construct the external database. Each generic image has a size 481×321, and some examples of the test generic images are shown in Fig. 2. In the targeted database case, following [1], we use the FEI face dataset [54] which has 200 aligned frontal face images of different persons with neutral expressions. We randomly separate them into two non-overlapping subsets, where one is used as the test images, and the other as the database. Each subset has 100 face images of size 260×360, and some examples of the test face images are shown in Fig. 3. Note that each database image is partitioned into 9×9 patches with 1-pixel overlap. Consequently, the generic database has $240,000$ patches, while the targeted database has $148,500$ patches.

Similar to the plaintext-domain work on image denoising [1], [6], [8], [35], we generate noisy images by adding zero-mean Gaussian noise with standard deviations ranging from $\sigma=10$ to $\sigma=40$ to the test images. Besides, we use two widely adopted metrics to evaluate the objective denoising quality, namely peak signal to noise ratio (PSNR) and structural similarity (SSIM) [55]. In particular, the PSNR measures the intensity difference between two images, while the SSIM measures the perceptual quality [56]. During denoising, an input noisy image is first partitioned into 9×9 patches with 1-pixel overlap, and then each patch is denoised separately. And following [8], we set the similarity threshold as $\varepsilon=k^2\times\sigma^2\times n$, where n is the number of pixels in a patch, i.e., 81 in our setting, and $k=1.126$. The decay parameter h is set to 0.35σ. All denoised patches are combined to reconstruct the denoised image, where the overlapping regions are averaged.

[2]OpenCV Library: http://opencv.org/

[3]Paillier Library:http://www.csee.umbc.edu/~kunliu1/research/Paillier.html

[4]https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

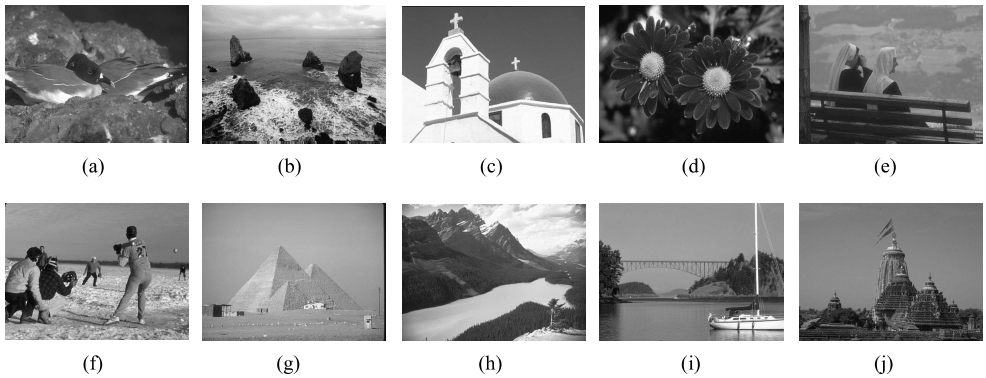


Fig. 2. Examples of the test generic images used in our experiments. They are denoted by "a", "b",..., "j" for convenience.

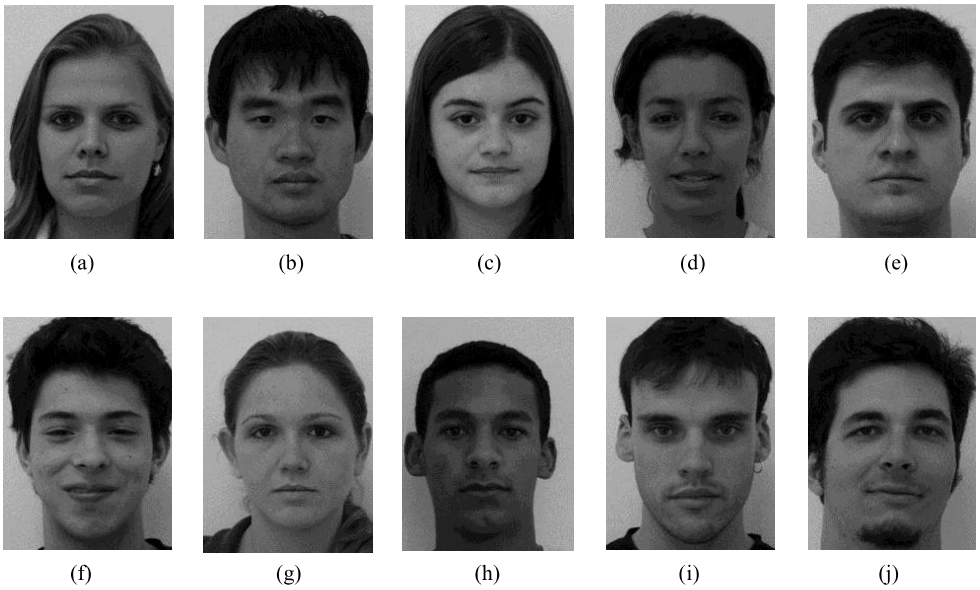


Fig. 3. Examples of the test face images used in our experiments. They are denoted by "a", "b",..., "j" for convenience.

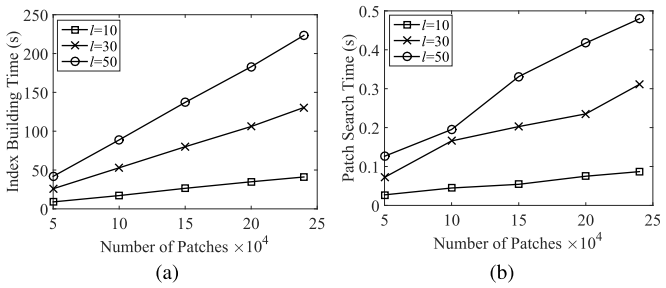


Fig. 4. The performance of the secure index. (a) The index building cost; (b) The patch search cost.

B. Performance Evaluation

1) Computation Performance: We measure the computation performance with regard to the secure index in our design. Fig. 4 shows the computation cost of index building and encrypted patch search. From Fig. 4(a), it can be observed that the index building time is linear in the size of the patch database. For l=10, it takes about 41 seconds to insert 240, 000 patches on average. When the number of LSH functions increases, the index building cost rises accordingly, say from 130.5 seconds to 223.4 seconds when l changes from 30 to 50. As for the patch search cost at the cloud, it also grows linearly in the number of database patches, as shown in Fig. 4(b). Besides, when l increases, the patch search cost grows accordingly. It takes about 203 μs to generate the search token for a query patch when $l = 10$, which is quite efficient. Moreover, it takes about 1.85 seconds to evaluate the squared Euclidean distance between the query patch and a candidate patch, including the time of circuit generation and evaluation.

2) Storage Performance: We report the storage consumption at the cloud side to support secure image external denoising in our design. Recall that the database primarily contains a secure index and patch ciphertexts. For the encrypted index, we use the standard HashMap in *Java* to store $l \times N$ key-value pairs, where N is the number of database patches. Given the default load factor 0.75, in the case of targeted database of 148, 500 patches, the index size is around 83.08 MB; in the case of generic database of 240, 000 patches, the index size is around 134.28 MB. As for the patch ciphertexts,

TABLE I

GENERIC DATABASE: COMPARISON OF PSNR (IN dB) AND SSIM RESULTS FOR 10 EXAMPLE TEST GENERIC IMAGES SHOWN IN FIG. 2

Image	Method	σ=10		σ=20		σ=30		σ=40	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
a	Baseline	30.103	0.774	23.476	0.492	22.254	0.443	21.598	0.386
	Secure LSH-voting	29.592	0.744	22.43	0.412	20.146	0.341	18.941	0.303
	Proposed	**30.038**	**0.773**	**23.101**	**0.466**	**21.381**	**0.417**	**20.119**	**0.355**
b	Baseline	29.841	0.846	23.3	0.645	20.916	0.552	19.561	0.465
	Secure LSH-voting	29.532	0.825	22.644	0.587	19.976	0.504	18.409	0.435
	Proposed	**29.779**	**0.843**	**23.079**	**0.629**	**20.513**	**0.542**	**18.975**	**0.461**
c	Baseline	32.189	0.828	24.958	0.583	22.948	0.503	21.909	0.434
	Secure LSH-voting	31.314	0.799	23.701	0.513	20.987	0.431	19.651	0.391
	Proposed	**31.71**	**0.824**	**24.403**	**0.564**	**22.106**	**0.49**	**20.58**	**0.433**
d	Baseline	30.647	0.805	24.482	0.581	23.502	0.536	22.684	0.464
	Secure LSH-voting	30.138	0.777	23.469	0.515	21.692	0.46	20.742	0.405
	Proposed	**30.592**	**0.804**	**24.216**	**0.57**	**22.807**	**0.52**	**21.795**	**0.446**
e	Baseline	31.1	0.778	23.969	0.461	22.873	0.414	22.43	0.37
	Secure LSH-voting	30.339	0.731	22.738	0.373	20.599	0.319	19.32	0.287
	Proposed	**30.945**	**0.772**	**23.494**	**0.433**	**21.839**	**0.39**	**20.829**	**0.346**
f	Baseline	30.094	0.839	22.9	0.554	20.771	0.456	20.144	0.397
	Secure LSH-voting	29.76	0.815	22.31	0.502	19.53	0.391	18.048	0.333
	Proposed	**30.019**	**0.837**	**22.684**	**0.538**	**20.216**	**0.438**	**18.92**	**0.367**
g	Baseline	31.22	0.797	23.548	0.44	21.861	0.356	21.61	0.3
	Secure LSH-voting	30.583	0.759	22.527	0.354	19.906	0.263	18.715	0.237
	Proposed	**31.053**	**0.793**	**23.177**	**0.413**	**21.048**	**0.341**	**20.166**	**0.291**
h	Baseline	30.187	0.776	23.807	0.523	22.632	0.471	22.175	0.42
	Secure LSH-voting	29.797	0.754	22.889	0.464	20.814	0.398	20.148	0.37
	Proposed	**30.079**	**0.771**	**23.447**	**0.505**	**21.822**	**0.454**	**21.123**	**0.407**
i	Basline	30.86	0.802	23.596	0.483	21.989	0.404	21.522	0.358
	Secure LSH-voting	30.367	0.771	22.594	0.409	20.029	0.318	18.738	0.28
	Proposed	**30.728**	**0.798**	**23.259**	**0.466**	**21.095**	**0.384**	**20.156**	**0.341**
j	Basline	31.237	0.834	23.443	0.487	21.868	0.403	21.138	0.328
	Secure LSH-voting	30.622	0.793	22.575	0.422	20.154	0.337	18.511	0.269
	Proposed	**31.082**	**0.831**	**23.16**	**0.474**	**21.208**	**0.4**	**19.706**	**0.311**

the size is $N \times (n + a)$ bytes, where n is the number of pixels in a patch, and a is the size of one Paillier ciphertext. Obviously, the size of patch ciphertexts is linear in the number of patches in the database. In our experiments, for $n = 81$ and $a = 256$ bytes, in the case of targeted database, this part consumes 40.59 MB; in the case of generic database, this part consumes 77.13 MB. Overall, our design incurs modest storage consumption at the cloud side to support secure image external denoising service.

3) Bandwidth Performance: We now report the bandwidth consumed for the user to query the encrypted cloud database for image denoising. Recall that for each noisy patch, the user needs to send a secure search token and one Paillier ciphertext to the cloud. Each search token contains $2 \times l$ sub-tokens, each of which has a size of 20 bytes via HMAC-SHA1. Therefore, the total size for each noisy query patch is $40 \times l + a$ bytes, where a is the size of one Paillier ciphertext. For $l = 10$ and $a = 256$ bytes, the bandwidth cost is only 0.64 KB for one query per patch, which is modest.

4) Denoising Quality: We evaluate the denoising quality in the cases of generic database and targeted database, respectively. For each kind of database, we evaluate our proposed method, and compare the result with a baseline method and a secure LSH-voting based method. Specifically, the baseline method is plaintext-domain external denoising that represents the optimal performance, where we denoise a noisy query patch by using accurately similar patches from a set of 60 database patches that have the smallest distances from it. As for the secure LSH-voting based method, we use accurately similar patches from the set of top-60 ranked candidate patches which are decided via the LSH-voting mechanism, for the denoising of a query patch. Recall that in our proposed method, we also leverage the LSH-voting mechanism for fast yet coarse-grained candidate patch selection, but further design a secure two-party computation protocol to filter the false-positive candidate patches. This ensures that similar patches, the target number of which is 60, are accurately obtained for denoising.

Table I shows the PSNR and SSIM results in the case of generic database, for the examples of test images displayed in Fig. 2. Further, Table II shows the average PSNR and SSIM results for the experiments on 100 test generic images. It is observed that the PSNR of the baseline method is higher than the proposed method, by 0.523 dB on average. Regarding the SSIM, the baseline method is 0.014 better than the proposed method. On another hand, compared with the secure LSH-voting based method, the PSNR and SSIM results of the proposed method are superior for all the noise levels. On average, in the generic database case, the proposed method achieves 0.643 dB higher PSNR and 0.036 higher SSIM.

The denoising quality results under different number of LSH functions is also compared in Table III for demonstration, using σ=20 as an example. It is shown that the denoising quality provided by our proposed method and the secure LSH-voting method could increase as more LSH functions are used. Meanwhile, even more LSH functions are used, the

TABLE II
GENERIC DATABASE: COMPARISON OF AVERAGE PSNR (IN dB) AND SSIM RESULTS OF 100 TEST GENERIC IMAGES

Image	Method	σ=10		σ=20		σ=30		σ=40	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ave.	Baseline	30.165	0.821	23.263	0.556	21.551	0.48	20.892	0.418
	Secure LSH-voting	29.776	0.799	22.548	0.505	20.067	0.412	18.816	0.359
	Proposed	**30.077**	**0.819**	**23.009**	**0.542**	**20.898**	**0.461**	**19.796**	**0.398**

TABLE III
VARYING LSH PARAMETER L: AVERAGE PSNR (IN dB) AND SSIM RESULTS OF 100 TEST GENERIC IMAGES (WITH $\sigma = 20$)

Image	Method	l=10		l=30		l=50	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ave.	Baseline	23.263	0.556	23.263	0.556	23.263	0.556
	Secure LSH-voting	22.548	0.505	22.77	0.524	22.873	0.532
	Proposed	**23.009**	**0.542**	**23.084**	**0.547**	**23.115**	**0.55**

denoising quality of our proposed method still outperforms the secure LSH-voting method. Such quality gain is promised as our design introduces a secure cloud-side filtering step to filter false-positive candidate patches caused by the inherent approximation nature of LSH.

We further evaluate the visual denoising quality in the case of generic database. Fig. 5 shows the visual denoising results under different methods for the generic image shown in Fig. 2(a) when σ=10. To show the details clearly, we highlight two cropped regions with white rectangles and show them at the bottom of the corresponding image. It is observed that the details produced by the proposed method and the baseline method are close. Meanwhile, compared with the secure LSH-voting method, the proposed method produces the details that are much closer to the original image. Hence, the proposed method also outperforms the secure LSH-voting method in terms of visual denoising quality.

Table IV shows the PSNR and SSIM results in the case of targeted database, for the examples of test face images displayed in Fig. 3. Further, Table V shows the average PSNR and SSIM results for the experiments on 100 test face images. And an example of visual denoising result is illustrated in Fig. 6. It can be observed that the baseline method achieves higher PSNR values than the proposed method, by 0.516 dB on average. In terms of the SSIM, the baseline method achieves higher values than the proposed method in most cases, but sometimes the proposed method is better. The reason is that sometimes the distance measuring the intensity difference might fail to reflect the structure similarity faithfully. Thus, the patches adopted by the proposed method for denoising might have structures more similar to the original image. Compared with the secure LSH-voting based method, the proposed method is superior in terms of both the PSNR and SSIM for all the noise levels, though the denoising results are visually close. On average, in the case of targeted database, the proposed method achieves 0.179 dB higher PSNR and 0.004 higher SSIM. Such improvement is less significant than that in the generic database case, because targeted database consists of images relevant to the noisy images only [1]. Therefore, the capability of LSH in locating similar patches for denoising could be superior to that in the generic database case, which degrades the significance of secure patch evaluation. On another hand, it should be noted that image denoising from generic external database is the general case in practice.

VII. RELATED WORK

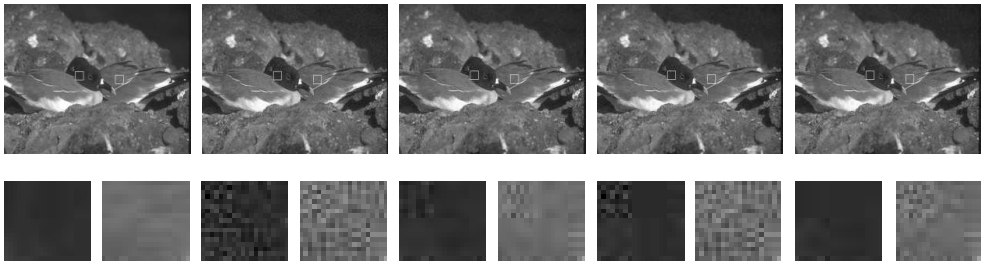
A. Image Denoising From Plaintext External Databases

Image denoising from external databases has become popular in recent years. Levin et al. [14] show that in theory the minimum mean squared error of denoising is achievable by using an infinitely large external database. Chan et al. [6] propose a computationally efficient random sampling scheme for patch selection, reducing the complexity of using large databases. These works use generic databases. Very recently, Luo et al. [1] studied image denoising from targeted databases. They apply a group sparsity minimization and a localized prior to learn the optimal denoising filter. Although effective, all the above works operate in the plaintext domain.

B. Similarity Search Over Encrypted Data

Our proposed design is also akin to the works on encrypted similarity search (to just list a few). In [18], Kuzu et al. study similarity search over encrypted high-dimensional data. They leverage LSH and build a secure index based on inverted index. Their design, however, suffers from space inefficiency and index imbalance. In [19], Yuan et al. propose a secure similarity index design for low latency applications where each query only needs to retrieve a small number of similar records with a controllable tradeoff on accuracy. In [57], Cui et al. propose a secure cloud-assisted mobile image sharing system, leveraging image correlation to support secure image reproduction at cloud. They build their secure similarity index over SIFT features. In these works, the search result contains false positives.

In this paper, we initiate the first study for privacy-preserving image denoising from external cloud databases. Our design leverages the very recent high performance SSE construction [43] and integrates it with LSH, enabling a user to securely locate candidate patches for denoising. In order to obtain high quality patches that satisfy the distance metric



(a) (b) (c) (d) (e)

Fig. 5. Generic database: example of visual denoising results when $\sigma = 10$. (a) Original image; (b) Noisy image; (c) Baseline result; (d) Secure LSH-voting result; (e) Our result.

TABLE IV

TARGETED DATABASE: COMPARISON OF PSNR (IN dB) AND SSIM RESULTS FOR 10 EXAMPLE TEST FACE IMAGES AS SHOWN IN FIG. 3

Image	Method	σ=10		σ=20		σ=30		σ=40	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
a	Baseline	35.774	0.911	32.52	0.836	30.19	0.754	27.997	0.665
	Secure LSH-voting	35.012	0.9	31.576	0.822	29.48	0.749	27.232	0.668
	Proposed	**35.103**	**0.903**	**31.725**	**0.824**	**29.628**	**0.751**	**27.367**	**0.67**
b	Baseline	36.225	0.915	32.509	0.835	29.696	0.739	27.379	0.646
	Secure LSH-voting	35.619	0.906	31.792	0.826	29.211	0.748	26.971	0.661
	Proposed	**35.639**	**0.908**	**32.024**	**0.83**	**29.523**	**0.754**	**27.12**	**0.664**
c	Baseline	36.239	0.914	32.688	0.839	29.941	0.751	27.706	0.671
	Secure LSH-voting	35.237	0.9	32.031	0.827	29.402	0.751	27.31	0.672
	Proposed	**35.487**	**0.904**	**32.275**	**0.832**	**29.694**	**0.758**	**27.379**	**0.675**
d	Baseline	36.336	0.914	32.437	0.828	29.819	0.732	27.645	0.644
	Secure LSH-voting	35.513	0.907	31.762	0.818	29.312	0.734	27.038	0.653
	Proposed	**35.719**	**0.91**	**32.059**	**0.826**	**29.541**	**0.739**	**27.241**	**0.66**
e	Baseline	36.401	0.919	32.578	0.846	29.875	0.757	27.783	0.674
	Secure LSH-voting	35.494	0.908	31.928	0.838	29.334	0.761	27.35	0.688
	Proposed	**35.642**	**0.911**	**32.137**	**0.843**	**29.682**	**0.769**	**27.484**	**0.69**
f	Baseline	36.263	0.915	32.45	0.841	29.992	0.76	27.844	0.674
	Secure LSH-voting	35.554	0.907	31.865	0.831	29.382	0.753	27.518	0.688
	Proposed	**35.594**	**0.909**	**32.095**	**0.836**	**29.617**	**0.76**	**27.657**	**0.69**
g	Baseline	36.281	0.911	32.879	0.837	30.468	0.754	28.56	0.674
	Secure LSH-voting	35.322	0.898	31.815	0.821	29.678	0.752	27.451	0.666
	Proposed	**35.485**	**0.902**	**32.111**	**0.826**	**29.918**	**0.76**	**27.717**	**0.673**
h	Baseline	36.748	0.919	32.74	0.832	30.338	0.745	28.106	0.656
	Secure LSH-voting	35.948	0.913	32.064	0.829	29.736	0.75	27.621	0.663
	Proposed	**36.044**	**0.914**	**32.267**	**0.834**	**29.923**	**0.754**	**27.775**	**0.667**
i	Baseline	36.089	0.91	32.577	0.836	30.09	0.748	27.964	0.669
	Secure LSH-voting	35.252	0.898	31.738	0.823	29.375	0.743	27.344	0.672
	Proposed	**35.35**	**0.901**	**31.925**	**0.829**	**29.605**	**0.745**	**27.441**	**0.672**
j	Baseline	36.169	0.913	32.483	0.837	29.898	0.75	27.564	0.655
	Secure LSH-voting	35.288	0.901	31.791	0.829	29.23	0.749	27.166	0.665
	Proposed	**35.473**	**0.905**	**32.034**	**0.832**	**29.419**	**0.752**	**27.3**	**0.669**

(a) (b) (c) (d) (e)

Fig. 6. Targeted database: example of visual denoising results when $\sigma = 10$. (a) Original image; (b) Noisy image; (c) Baseline result; (d) Secure LSH-voting result; (e) Our result.

for denoising, we further design and implement a secure two-party computation protocol based on Yao's garbled circuits to filter the false positives at the cloud. Note that we resort to the SSE construction in [43] mainly due to its high performance in index size, search computation, etc. Meanwhile, it can be implemented via generic dictionary structure and thus has good support for practical deployment. A survey on secure searchable encryption can be found in [58], which provides a comprehensive overview of both SSE and public key searchable encryption (PKSE). Compared with SSE, PKSE techniques are usually very computationally expensive [59].

TABLE V
TARGETED DATABASE: COMPARISON OF AVERAGE PSNR (IN dB) AND SSIM RESULTS OF 100 TEST FACE IMAGES

Image	Method	σ=10		σ=20		σ=30		σ=40	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Ave.	Baseline	36.11	0.915	32.377	0.838	29.82	0.75	27.678	0.665
	Secure LSH-voting	35.329	0.905	31.603	0.827	29.166	0.748	27.108	0.671
	Proposed	**35.456**	**0.908**	**31.829**	**0.832**	**29.375**	**0.753**	**27.263**	**0.674**

C. Secure Computation Exploiting Garbled Circuits

Our work is also related to the works (to just list a few) using Yao's garbled circuits for secure computation. In [60], Huang et al. consider the privacy-preserving biometric identification scenario where a user interactively runs the identification protocol with the server hosting a biometric database, so as to securely find the closest match. They exploit garbled circuits to securely evaluate which database fingerprint has the minimum distance from the query fingerprint. Yao's garbled circuits are also employed for privacy-preserving ridge-regression in [45] and for privacy-preserving matrix factorization in [23], respectively. These two works both assume the existence of a third party generating garbled circuits, who is assumed not to collude with the evaluator that collects user's encrypted data. Similar to [45] and [23], our design also assumes the existence of a non-colluding extra server which assists the cloud to securely evaluate the located candidate patches, ensuring that similar patches are accurately obtained for image denoising.

VIII. CONCLUSION

We have initiated the first endeavor toward privacy-preserving image denoising from external cloud databases. Our proposed design enables the cloud hosting encrypted databases to offer secure query-based image denoising services. Leveraging the encrypted similarity search bridging SSE and LSH as our starting point, we have designed and implemented a secure computation protocol based on Yao's garbled circuits to ensure that similar patches are accurately obtained for promising denoising performance. Formal security analysis has been provided to justify the security guarantees of our design, and extensive experiments over real-world datasets have demonstrated that our design can achieve the denoising quality close to the optimal performance in plaintext.

REFERENCES

[1] E. Luo, S. H. Chan, and T. Q. Nguyen, "Adaptive image denoising by targeted databases," *IEEE Trans. Image Process.*, vol. 24, no. 7, pp. 2167–2181, Jul. 2015.

[2] H. Yue, X. Sun, J. Yang, and F. Wu, "Image denoising by exploring external and internal correlations," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1967–1982, Jun. 2015.

[3] A. Wong, A. Mishra, W. Zhang, P. Fieguth, and D. A. Clausi, "Stochastic image denoising based on Markov-chain Monte Carlo sampling," *Signal Process.*, vol. 91, no. 8, pp. 2112–2120, 2011.

[4] P. Chatterjee and P. Milanfar, "Patch-based near-optimal image denoising," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1635–1649, Apr. 2012.

[5] C. Sutour, C.-A. Deledalle, and J.-F. Aujol, "Adaptive regularization of the NL-means: Application to image and video denoising," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3506–3521, Aug. 2014.

[6] S. H. Chan, T. Zickler, and Y. M. Lu, "Monte Carlo non-local means: Random sampling for large-scale image filtering," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3711–3725, Aug. 2014.

[7] E. Luo, S. H. Chan, and T. Q. Nguyen, "Image denoising by targeted external databases," in *Proc. IEEE ICASSP*, May 2014, pp. 2450–2454.

[8] C. Lee, C. Lee, and C.-S. Kim, "An MMSE approach to nonlocal image denoising: Theory and practical implementation," *J. Vis. Commun. Image Represent.*, vol. 23, no. 3, pp. 476–490, 2012.

[9] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[10] P. Mell and T. Grance. (2011). *The NIST Definition of Cloud Computing*. [Online]. Available: http://csrc.nist.gov/publications/PubsSPs.html#800-145

[11] H. Yue, X. Sun, J. Yang, and F. Wu, "Cloud-based image coding for mobile devices—Toward thousands to one compression," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 845–857, Jun. 2013.

[12] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, "Privacy-assured outsourcing of image reconstruction service in cloud," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 1, pp. 166–177, Jun. 2013.

[13] S. Wang, K. Gu, S. Ma, W. Lin, X. Liu, and W. Gao, "Guided image contrast enhancement based on retrieved images in cloud," *IEEE Trans. Multimedia*, vol. 18, no. 2, pp. 219–232, Feb. 2016.

[14] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Proc. IEEE CVPR*, Jun. 2011, pp. 2833–2840.

[15] A. L. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Proc. ECCV*, 2012, pp. 73–86.

[16] BBC. (2014). *Apple to Tighten iCloud Security After Celebrity Leaks*. [Online]. Available: http://www.bbc.com/news/technology-29076899

[17] North Carolina Daily. (2014). *Snapchat Nude Photos, Videos Reportedly Leaked Online*. [Online]. Available: http://www.northcarolinadaily.com/index.php/sid/226634683

[18] M. Kuzu, M. Islam, and M. Kantarcioglu, "Efficient similarity search over encrypted data," in *Proc. IEEE ICDE*, Apr. 2012, pp. 1156–1167.

[19] X. Yuan, H. Cui, X. Wang, and C. Wang, "Enabling privacy-assured similarity retrieval over millions of encrypted records," in *Proc. ESORICS*, 2015, pp. 40–60.

[20] X. Yuan, X. Wang, C. Wang, J. Weng, and K. Ren, "Enabling secure and fast indexing for privacy-assured healthcare monitoring via compressive sensing," *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2002–2014, Oct. 2016.

[21] L. Zhang, T. Jung, C. Liu, X. Ding, X.-Y. Li, and Y. Liu, "Pop: Privacy-preserving outsourced photo sharing and searching for mobile devices," in *Proc. IEEE ICDCS*, Jul. 2015, pp. 308–317.

[22] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *Proc. ACM CCS*, 2013, pp. 247–258.

[23] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proc. ACM CCS*, 2013, pp. 801–812.

[24] S. Kim, J. Kim, D. Koo, Y. Kim, H. Yoon, and J. Shin, "Efficient privacy-preserving matrix factorization via fully homomorphic encryption," in *Proc. ACM ASIACCS*, 2016, pp. 617–628.

[25] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Enabling encrypted cloud media center with secure deduplication," in *Proc. ACM ASIACCS*, 2015, pp. 63–72.

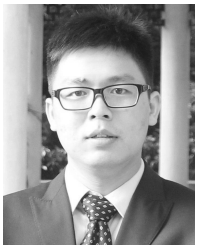
[26] S. Hu, Q. Wang, J. Wang, Z. Qin, and K. Ren, "Securing SIFT: Privacy-preserving outsourcing computation of feature extractions over encrypted image data," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3411–3425, Jul. 2016.

[27] Z. Qin, J. Yan, K. Ren, C. W. Chen, and C. Wang, "Secsift: Secure image SIFT feature extraction in cloud computing," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 12, no. 4s, pp. 65-1–65-24, 2016.

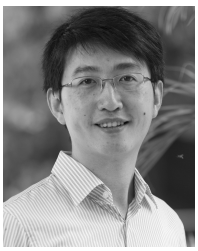
[28] Q. Wang, J. Wang, S. Hu, Q. Zou, and K. Ren, "Sechog: Privacy-preserving outsourcing computation of histogram of oriented gradients in the cloud," in *Proc. ACM ASIACCS*, 2016, pp. 257–268.

[29] H. Cui, X. Yuan, Y. Zheng, and C. Wang, "Enabling secure and effective near-duplicate detection over encrypted in-network storage," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[30] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. IEEE FOCS*, Oct. 1986, pp. 162–167.
[31] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "ObliVM: A programming framework for secure computation," in *Proc. IEEE SP*, May 2015, pp. 359–376.
[32] T. S. Cho, S. Avidan, and W. T. Freeman, "A probabilistic image jigsaw puzzle solver," in *Proc. IEEE CVPR*, 2010, pp. 183–190.
[33] D. Pomeranz, M. Shemesh, and O. Ben-Shahar, "A fully automated greedy square jigsaw puzzle solver," in *Proc. IEEE CVPR*, Jun. 2011, pp. 9–16.
[34] H. Chun, Y. Elmehdwi, F. Li, P. Bhattacharya, and W. Jiang, "Outsourceable two-party privacy-preserving biometric authentication," in *Proc. ACM ASIACCS*, 2014, pp. 401–412.
[35] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
[36] Q. Guo, C. Zhang, Y. Zhang, and H. Liu, "An efficient SVD-based method for image denoising," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 5, pp. 868–880, May 2016.
[37] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE CVPR*, Jun. 2005, pp. 60–65.
[38] S. Pyatykh, J. Hesser, and L. Zheng, "Image noise level estimation byprincipal component analysis," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 687–699, Feb. 2013.
[39] X. Liu, M. Tanaka, and M. Okutomi, "Single-image noise level estimation for blind denoising," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5226–5237, Dec. 2013.
[40] C. Tang, X. Yang, and G. Zhai, "Noise estimation of natural images via statistical analysis and noise injection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1283–1294, Aug. 2015.
[41] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for two-party computation," *J. Cryptol.*, vol. 22, no. 2, pp. 161–188, Apr. 2009.
[42] 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.
[43] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," in *Proc. NDSS*, 2014, pp. 1–16.
[44] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. ACM SOSP*, 2011, pp. 85–100.
[45] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE SP*, May 2013, pp. 334–348.
[46] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999, pp. 518–529.
[47] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM CCS*, 2012, pp. 965–976.
[48] X. Yuan, X. Wang, C. Wang, A. Squicciarini, and K. Ren, "Enabling privacy-preserving image-centric social discovery," in *Proc. IEEE ICDCS*, Jul. 2014, pp. 198–207.
[49] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. ACM SOSP*, 2011, pp. 85–100.
[50] K. Vu, R. Zheng, and J. Gao, "Efficient algorithms for k-anonymous location privacy in participatory sensing," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2399–2407.
[51] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.
[52] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li, "Modeling LSH for performance tuning," in *Proc. ACM CIKM*, 2008, pp. 669–678.
[53] M. Slaney, Y. Lifshits, and J. He, "Optimal parameters for locality-sensitive hashing," *Proc. IEEE*, vol. 100, no. 9, pp. 2604–2623, Sep. 2012.
[54] C. E. Thomaz and G. A. Giraldi, "A new ranking method for principal components analysis and its application to face image analysis," *Image Vis. Comput.*, vol. 28, no. 6, pp. 902–913, 2010.
[55] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[56] D.-H. Trinh, M. Luong, F. Dibos, J.-M. Rocchisani, C.-D. Pham, and T. Q. Nguyen, "Novel example-based method for super-resolution and denoising of medical images," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1882–1895, Apr. 2014.
[57] H. Cui, X. Yuan, and C. Wang, "Harnessing encrypted data in cloud for secure and efficient image sharing from mobile devices," in *Proc. IEEE INFOCOM*, May 2015, pp. 2659–2667.
[58] C. Bösch, P. H. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 18-1–18-51, 2015.
[59] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[60] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," in *Proc. NDSS*, pp. 1–14, 2011.

Yifeng Zheng received the B.E. degree in information engineering from the South China University of Technology, Guangzhou, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong. In 2013, he was with Zhejiang University, Hangzhou, China. His research interests include cloud security, encrypted computation, and multimedia security.



Helei Cui received the B.E. degree in software engineering from Northwestern Polytechnical University, Xi'an, China, in 2010, and the M.S. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include cloud computing security, mobile computing security, and multimedia security.



Cong Wang received the B.E. and M.E. degrees from Wuhan University, Wuhan, China, in 2004 and 2007, respectively, and the Ph.D. degree from the Illinois Institute of Technology, Chicago, IL, USA, in 2012, all in electrical and computer engineering. He is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. He was with the Palo Alto Research Center in the summer of 2011. His research interests include cloud security, big data, and multimedia security.

Jiantao Zhou received the B.E. degree from the Dalian University of Technology, Dalian, China, in 2002, the M.Phil. degree from Southeast University, Nanjing, China, in 2005, and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2009. He held various research positions with the University of Illinois at Urbana-Champaign, Hong Kong University of Science and Technology, and McMaster University. He is currently an Assistant Professor with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau. He holds three granted U.S. patents and two granted Chinese patents. His research interests include multimedia security and forensics, and high-fidelity image compression. He was a co-author of two papers that received the Best Paper Award at the IEEE Pacific-Rim Conference on Multimedia (PCM) in 2007, and the Best Student Paper Award at the IEEE International Conference on Multimedia and Expo (ICME) in 2016, respectively.