SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

June 2017, Vol. 60 062101:1–062101:17
doi: 10.1007/s11432-016-0242-7

hOPE: improved order preserving encryption with the power to homomorphic operations of ciphertexts

Yanguo PENG[1], Hui LI[2], Jiangtao CUI[1*], Junwei ZHANG[2], Jianfeng MA[2] & Changgen PENG[3]

[1]*School of Computer Science and Technology, Xidian University, Xi'an* 710071*, China;*
[2]*School of Cyber Engineering, Xidian University, Xi'an* 710071*, China;*
[3]*College of Science, Guizhou University, Guiyang* 550025*, China*

Received July 28, 2016; accepted September 8, 2016; published online January 21, 2017

Abstract Database applications that manage and utilize massive data must address the issues of element comparisons, the core operations in index accessing, metric computations and metric comparisons, and the core operations in result ranking. In the cloud era, to avoid private information leakage, encrypted data are subcontracted, and resolutions for the problems that arise in three operations over ciphertexts are urgently required. Indeed, it is possible to handle element comparison through order preserving encryption/encoding (OPE) or metric computation through homomorphic encryption (HE) directly over ciphertexts. Unfortunately, the simultaneous achievement of both goals (i.e., metric computation and comparison) by directly combining OPE and HE remains intractable. In this work, an improved OPE, named hOPE, is proposed to support homomorphic operations over ciphertexts in addition to comparisons. Based on hOPE, AhOPE and PhOPE are designed to support homomorphic addition and product, respectively. Both schemes are proved to be indistinguishable under operated and ordered chosen-plaintext attack (IND-O^{2}CPA) secure when the adopted HE algorithm provides indistinguishable under ordered chosen-plaintext attack (IND-OCPA secure). hOPE is a general construction that supports arbitrary HE algorithms and achieves consistent security. We deploy AhOPE and PhOPE in practice with a trusted/untrusted third party and compare the result with the state-of-the-art methods. The results show that our presented algorithms need few interactions and fill the gap between OPE and HE.

Keywords order preserving encoding, homomorphic operation, B$^+$-tree, lookup table, provable secure

Citation Peng Y G, Li H, Cui J T, et al. hOPE: improved order preserving encryption with the power to homomorphic operations of ciphertexts. Sci China Inf Sci, 2017, 60(6): 062101, doi: 10.1007/s11432-016-0242-7

1 Introduction

In order to efficiently handle various queries over massive data in data-intensive applications, an index is essential for quickly retrieving data items that closely match a given query. As a typical and core instance in data management and utilization, retrieval is divided into two phases: first, the candidates are located by index accessing (i.e., filter), and then ranked results are obtained by pruning redundant candidates (i.e., refine). This process faces three inevitable issues, element comparison in filter, metric computation

* Corresponding author (email: cuijt@xidian.edu.cn)

and metric comparison in refine. Driven by the overwhelming amount of data and corresponding computations, organizations are subcontracting encrypted copies of data (i.e., plaintext) to a public cloud, which reduces consumer's hardware investment and maintenance cost[1)]. However, the encrypted data cannot be indexed or utilized directly, since the encryption breaks the semantics of data. Thus, effective mechanisms are urgently required for encrypted data (i.e., ciphertext) to resolve the above issues that raise concern in the cloud computing era.

To resolve element comparison (the first issue) over ciphertexts, order preserving encryption/encoding (OPE) [1] was introduced. By encrypting elements in the index using OPE, a public cloud is able to index the encrypted data and quickly locate response candidates for a query, such as a k-nearest neighbor (kNN) query or a range query. In general, OPE satisfies the following properties.

• **Order preserving.** The ciphertexts preserve the ordering of the plaintexts. This means that, for two arbitrary plaintexts $x < y$, the order relation $E(x) < E(y)$ between ciphertexts is preserved.

• **Decryption ability.** There exists an inverse (decryption) algorithm for the encryption algorithms. This means that, for any arbitrary ciphertext $E(x)$, plaintext x can be derived in polynomial time.

A large number of studies have been conducted [2–6] for essentially improving the efficiency and security of OPE. Popa et al. [3] proposed mutable OPE (mOPE) with ideal security, which is indistinguishability under ordered chosen-plaintext attack (IND-OCPA) secure. Most other solutions leak more than only the ordering of the data, as stated in [3]. Although IND-OCPA security has been satisfied, none of the current OPE solutions provides a feasible approach for supporting homomorphic operations over ciphertexts. Hence, OPE solutions cannot resolve metric computation and comparison issues directly over ciphertexts. Because of these drawbacks, only a tiny part of indexes with element comparison can be realized without computations or comparisons of various boundaries, such as those realized by KD-tree [7] and B-tree [8]. Other index structures employing boundaries, such as k-means [9], R-tree [10], cannot be implemented directly based on OPE. Still, the process of refinement based on OPE is extremely difficult, because metric computation and comparison are employed.

To resolve metric computation (the second issue) directly over ciphertexts, homomorphic operations are fundamental and therefore it is urgent to solve this issue. The first encryption algorithm with this characteristic was homomorphic encryption (HE) [11]. Paillier [12] revisited HE and proposed the first HE algorithm with the ability to handle homomorphic addition at EUROCRYPT in 1999. Then, fully homomorphic encryption (FHE), such that homomorphic product can be preserved in ciphertexts, was investigated in depth in [13–15]. Briefly, the following properties are partially (resp. entirely) satisfied for HE (resp. FHE).

• **Homomorphic addition.** The encryption algorithm preserves the addition of plaintexts. This means that, for two arbitrary plaintexts x and y, $E(x + y) = E(x) + E(y)$ is preserved [2)].

• **Homomorphic product.** The encryption algorithm preserves the product of plaintexts. This means that, for two arbitrary plaintexts x and y, $E(x \times y) = E(x) \times E(y)$ is preserved.

• **Decryption ability.** An inverse (decryption) algorithm exists for the encryption algorithms. This means that, for any arbitrary ciphertext $E(x)$, plaintext x can be derived in polynomial time.

According to the above properties, numerous variants of metric computations can be implemented by specific approaches, such as those using squared Euclidean distance [16] and various boundaries.

Despite the fact that element comparison and metric computation (the first two issues) over ciphertexts can be resolved effectively by combining OPE and HE/FHE directly, metric comparison over ciphertexts is still intractable and challenging. The reason is that, the encrypted metric is finally obtained in the form of ciphertext in HE/FHE, while the order preserving ciphertexts in current OPE solutions are completely different. Obviously, there is a huge gap in that ciphertexts in HE/FHE cannot be compared directly, and ciphertexts in OPE cannot support homomorphic operations. In this study, our ultimate objective was

1) http://cloudtimes.org/2012/12/20/netflix-aws-2013/.

2) The addition "+" is an abstract representation. In fact, in concrete HE/FHE, the addition "+" is achieved by more than concrete additive numerical operations. The same applies to the product "×" in the following. For example, in Paillier [12], the homomorphic addition is achieved by multiplication of ciphertexts. Indeed, how to achieve concrete homomorphic operations beyond the scope of this paper.

to design an improved order preserving encryption algorithm with the power to support homomorphic operations of ciphertexts, called hOPE. Our proposed method is aimed to address all the three issues completely and finally break the bottleneck of security issues in the aforementioned scenarios. The most significant contributions of this paper are as follows.

• Having conducted very considerable investigations, we propose an ideal security model for hOPE, called indistinguishability under operated and ordered chosen-plaintext attack (IND-O^{2}CPA).

• We construct two concrete IND-O^{2}CPA secure hOPE schemes, AhOPE and PhOPE, to support homomorphic addition and product, respectively.

• We resolve several concerns that arose while deploying hOPEs. Additionally, our proposed models are proved to be superior to several other related methods.

In Section 2, we present the problem definition and fundamental preliminaries. Then, we formally describe the ideal model of IND-O^{2}CPA secure and several cryptographic tools for constructing a concrete hOPE. A concrete hOPE with the ability of homomorphic addition (AhOPE) is shown in Section 3. In addition, a concrete hOPE with the ability of homomorphic product (PhOPE) is presented in Section 4. In Section 5, we resolve several issues that arose when deploying hOPEs in practice, and show theoretical comparisons. Related work is presented in Section 6. Finally, we conclude this paper in Section 7.

2 Problem definition and preliminaries

We first formally define the hOPE model and state the urgent challenges. Then, we analyze the challenges in terms of security and derive an appropriate security model based on IND-OCPA. Finally, several necessary cryptographic tools for constructing a concrete hOPE are investigated in depth.

2.1 Model and challenges

Definition 1 (hOPE)**.** An hOPE scheme includes the following algorithms:

• **Setup.** Take as input a security parameter κ and return a system parameter SP. The system parameter additionally includes an HE/FHE scheme.

• **KeyGen.** Take as input SP and generate an encryption (public) key pk and a decryption (private) key sk for the HE/FHE scheme.

• **Enc.** Take as input SP, pk, and a plaintext m. Output the ciphertext cipher.

• **Dec.** Take as input SP, sk, and a ciphertext cipher. Output the plaintext m.

• **Homomorphic operations.** Take as input SP, and two ciphertexts cipher_1 and cipher_2. The following operations are performed partially or entirely, depending on the adopted HE/FHE scheme:

- **Homomorphic addition.** Output the sum $\text{cipher}_1 + \text{cipher}_2$ as the result.

- **Homomorphic product.** Output the product $\text{cipher}_1 \times \text{cipher}_2$ as the result.

• **Comp.** Take as input SP and two ciphertexts cipher_1 and cipher_2 for plaintexts m_1 and m_2, respectively. Output the comparison result between m_1 and m_2.

When deploying hOPE in practice, the ciphertexts are subcontracted to a semi-honest remote server. When a trusted client (owning the secret key) issues queries, the server can respond with the query result.

Definition 2 (Completeness)**.** An hOPE is complete iff the following characteristics are satisfied:

• **Decryption ability.** For an arbitrary ciphertext cipher, there exists an efficient algorithm for deriving the corresponding plaintext m.

• **Additivity.** If homomorphic addition is feasible in the adopted HE/FHE scheme, given arbitrary m_1, m_2, $\text{cipher}_1 = \text{Enc}(m_1)$, and $\text{cipher}_2 = \text{Enc}(m_2)$, $\text{Dec}(\text{cipher}_1 + \text{cipher}_2) = m_1 + m_2$.

• **Productivity.** If homomorphic product is feasible in the adopted HE/FHE scheme, given arbitrary m_1, m_2, $\text{cipher}_1 = \text{Enc}(m_1)$, and $\text{cipher}_2 = \text{Enc}(m_2)$, $\text{Dec}(\text{cipher}_1 \times \text{cipher}_2) = m_1 \times m_2$.

• **Comparability.** For arbitrary ciphertexts cipher_1 and cipher_2, $\text{cipher}_1 < \text{cipher}_2$ iff $m_1 < m_2$.

As compared with existing OPEs, hOPE allows homomorphic operations to be achieved directly. In [3], order preservation was achieved by transforming deterministic ciphertext to a unique order preserving

code. However, ciphertexts for HE/FHE are randomized and break the transformation. Additionally, the homomorphic operations will more or less affect the security. We discuss this later.

2.2 Cryptographic preliminaries

To explain the construct of hOPE, several necessary cryptographic preliminaries are now introduced.

Suppose $\mathbb{G}$ is an additive cyclic group and $\mathbb{H}$ is a multiplicative cyclic group. They both have the same order q. A bilinear pairing map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$ is constructed satisfying the following characteristics [17]:

- **Bilinear.** For any $P, Q \in \mathbb{G}$ and any $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(bP, aQ) = \hat{e}(P, Q)^{ab}$.
- **Non-degenerate.** There exists $P, Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1 \in \mathbb{H}$, where 1 is the identity element.
- **Computable.** For any $P, Q \in \mathbb{G}$, there is an efficient algorithm to compute $\hat{e}(P, Q) \in \mathbb{H}$.

Indeed, $\hat{e}$ is also symmetric since it is bilinear and $\mathbb{G}$ is a cyclic group. Moreover, there are several implicit assumptions, based on several public problems that a bilinear pairing map follows.

- **Elliptic curve discrete logarithm (ECDL) problem.** Given any different P, Q in $\mathbb{G}$ (resp. $\mathbb{H}$), compute $a \in Z_q^*$ such that $Q = aP$.
- **Computational Diffie-Hellman (CDH) problem.** Given $P, aP, bP \in \mathbb{G}$, compute $abP \in \mathbb{G}$.
- **Computational bilinear Diffie-Hellman (CBDH) problem.** Given any $P \in \mathbb{G}$, $\hat{e}$, as well as aP, bP and cP for any unknown $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc} \in \mathbb{H}$.
- **Bilinear pairing inversion (BPI) problem.** Given any $P \in \mathbb{G}$ and $\hat{e}(P, Q) \in \mathbb{H}$, compute $Q \in \mathbb{G}$.
- **Decisional Diffie-Hellman (DDH) problem.** Given any $P, aP, bP, cP \in \mathbb{G}$, in which $a, b, c \in \mathbb{Z}_q^*$ are selected randomly and unknown, judge whether $c = ab \mod q$ or not.

Bilinear pairing is prevalent in cryptographic protocols [17–20] with the following implicit assumptions in polynomial time.

- There is no efficient algorithm that can resolve the ECDL, CDH, CBDH, or BPI problem.
- There is an efficient algorithm that can resolve the DDH problem.

2.3 Security model

When constructing a concrete hOPE, an appropriate security model is of primary concern. Before stating our security definition for hOPE, we introduce the related auxiliary notions of which we make use. The allowed leakage in OPE is a relation pattern that exposes all the relations between plaintexts.

Definition 3 (Relation pattern)**.** Let $\mathbb{D}$ be a set of sortable plaintexts, where n is the size of $\mathbb{D}$ and x is the element in $\mathbb{D}$. The relation pattern is an $n \times n$ matrix $\upsilon(\mathbb{D}) = M$ such that for $1 \leqslant i, j \leqslant n$, $M_{ij} = -1$ if $x_i < x_j$, $M_{ij} = 0$ if $x_i = x_j$, and $M_{ij} = 1$ otherwise.

In existing IND-OCPA secure solutions [2, 3, 21], there is an assumption that the sortable set $\mathbb{D}$ is non-singular as defined below. The notation of non-singular is popular in related research fields [22–24]. Otherwise, the two challenged sets can be distinguished immediately.

Definition 4 (Non-singular set for OPE)**.** A sortable set $\mathbb{D}$ is non-singular if (1) there exists at least one set $\mathbb{D}' \neq \mathbb{D}$ such that $\upsilon(\mathbb{D}') = \upsilon(\mathbb{D})$ and (2) $\mathbb{D}'$ can be found in polynomial time given $\upsilon(\mathbb{D})$.

Security requirement analysis for hOPE. In fact, the above security model is not well suited for hOPE, because a more complicated concern arises when introducing a homomorphic operation $*$. Indeed, there is another assumption in HE/FHE that $\mathbb{D}$ is closed under the operation $*$ [12–14]. Hence, $x_i * x_j$ belongs to $\mathbb{D}$ if $x_i, x_j \in \mathbb{D}$. Otherwise, the homomorphic encryption cannot be continuously executed. Our objective is to suggest a new IND-O^{2}CPA security model, in which homomorphic operation $*$ is introduced on condition that the relation pattern of total plaintexts [3] is leaked. Such a leakage is called an extensional search pattern, as defined below.

Definition 5 (Extensional relation pattern)**.** Let $*$ be a homomorphic operation and $\mathbb{D}$ be a set of plaintexts, where n is the size of $\mathbb{D}$ and x is the element in $\mathbb{D}$. The closed set $\bar{\mathbb{D}}$ is generated by set $\mathbb{D}$

3) Here, $x_i * x_j$ is also included in $\mathbb{D}$, since $\mathbb{D}$ is closed under the operation $*$.

over the operation $*$. Obviously, $\bar{\mathbb{D}}$ is closed under the operation $*$. Assume that the size of $\bar{\mathbb{D}}$ is m. The extensional relation pattern under homomorphic operation $*$ for set $\mathbb{D}$ is an $m \times m$ matrix $\bar{v}(\mathbb{D}) = v(\bar{\mathbb{D}})$.

Obviously, in hOPE, only the extensional relation pattern is allowed to leak. The challenged sorted set $\mathbb{D}$ should also be non-regular, as defined below.

Definition 6 (Non-singular set for hOPE)**.** A sortable set $\mathbb{D}$ is non-singular if (1) there exists at least one set $\mathbb{D}' \neq \mathbb{D}$ such that $\bar{v}(\mathbb{D}') = \bar{v}(\mathbb{D})$ and (2) $\mathbb{D}'$ can be found in polynomial time given $\bar{v}(\mathbb{D})$.

Here, $\mathbb{D}$ is singular for OPE (resp. hOPE) if Definition 4 (resp. Definition 6) is not satisfied.

Theorem 1 (Impossibility)**.** Given a homomorphic operation $*$. An IND-OCPA (resp. IND-O^{2}CPA) secure OPE (resp. hOPE) algorithm E can be constructed only on condition that the challenged set $\mathbb{D}$ is non-singular for OPE (resp. hOPE).

Proof. We use proof by contraposition here. The contrapositive of the original proposition is that, if there is no IND-OCPA (resp. IND-O^{2}CPA) secure OPE (resp. hOPE) algorithm E, then $\mathbb{D}$ is singular for OPE (resp. hOPE).

Let $\mathbb{D}'$ be another sortable set. Both $\mathbb{D}$ and $\mathbb{D}'$ are challenged. In OPE (resp. hOPE), the leaked information is restricted only on (extensional) relation patterns $v(\mathbb{D})$ and $v(\mathbb{D}')$ (resp. $\bar{v}(\mathbb{D})$ and $\bar{v}(\mathbb{D}')$). The adversary can distinguish $E(\mathbb{D})$ and $E(\mathbb{D}')$, since there is no IND-OCPA (resp. IND-O^{2}CPA) secure OPE (resp. hOPE) algorithm E. Obviously, $v(\mathbb{D}) \neq v(\mathbb{D}')$ (resp. $\bar{v}(\mathbb{D}) \neq \bar{v}(\mathbb{D}')$) is derived, because of the restriction of the leaked information. This means $\mathbb{D}$ is singular according to Definition 4 (resp. Definition 6). The contraposition is proved, and hence, the original proposition is proved.

According to Theorem 1, it is impossible to construct secure OPE or hOPE without the restriction that the challenged set is non-singular. Furthermore, if the restriction is removed, a challenge on a singular sortable set is meaningless, specifically, if a singular set $\mathbb{D}$ is challenged together with another set $\mathbb{D}'$. Thus, both $\mathbb{D}$ and $\mathbb{D}'$ own different extensional relation patterns, i.e., $\bar{v}(\mathbb{D}) \neq \bar{v}(\mathbb{D}')$. The challenger returns $E(\Delta)$, where $\Delta = \mathbb{D}$ or $\Delta = \mathbb{D}'$, with probability $\Pr[\Delta = \mathbb{D}] = 1/2$. The adversary can immediately win the game by matching $\bar{v}(E(\Delta))$ with $\bar{v}(\mathbb{D})$ or $\bar{v}(\mathbb{D}')$, because $\bar{v}(E(\mathbb{D})) = \bar{v}(\mathbb{D})$ and $\bar{v}(E(\mathbb{D}')) = \bar{v}(\mathbb{D}')$. This winning of the game does not depend on a direct attack on the encryption algorithm. Hence, the adversary's attack is meaningless.

Security model. We define a new ideal security model, IND-O^{2}CPA. The challenge game operates as follows. First, the adversary submits arbitrary sets $\mathbb{X}$ and $\mathbb{Y}$ to the challenger with the restriction that $\bar{v}(\mathbb{X}) = \bar{v}(\mathbb{Y})$. Then, the challenger picks up a random coin $b \in \{0, 1\}$ such that $\Pr[b = 0] = 1/2$, and returns $E(\mathbb{X})$ if $b = 0$ and $E(\mathbb{Y})$ if $b = 1$. Finally, the adversary wins the game iff he guesses a bit b' such that $b' = b$. Note that, if the homomorphic operation has never been performed, IND-O^{2}CPA is obviously equivalent to IND-OCPA. Hence, IND-O^{2}CPA is an effective generalization of IND-OCPA.

Definition 7 (IND-O^{2}CPA secure)**.** In polynomial time, for two given encrypted sequences $E(\mathbb{X})$ and $E(\mathbb{Y})$ with the restriction that $\bar{v}(\mathbb{X}) = \bar{v}(\mathbb{Y})$, the encryption algorithm is IND-O^{2}CPA secure when the adversary's advantage $\mathrm{Adv}_{\mathcal{A}}$ distinguishing them is a negligible function. Formally,

$$\mathrm{Adv}_{\mathcal{A}} = \Pr[b' = b] - \frac{1}{2},$$

where $b' = b$ means that the adversary wins the challenge game.

By Definition 7, hOPE reveals nothing about ciphertexts, except the extensional relation pattern when IND-O^{2}CPA secure is satisfied. It is very practical and fundamental for the application scenarios described in Section 1. Here, we do not consider the full ciphertext space attack, since schemes against such attacks are not well promoted.

2.4 Code tree and improved lookup table

The first OPE algorithm satisfying IND-OCPA secure, mOPE, was proposed in [3], where a well-designed mechanism for transforming deterministic ciphertexts (e.g., AES and RC4) to order preserving codes based on B-tree and a lookup table was presented. There are two inevitable drawbacks: (1) The codes must be postpositively extended to a fixed-length, and (2) the original lookup table cannot transform

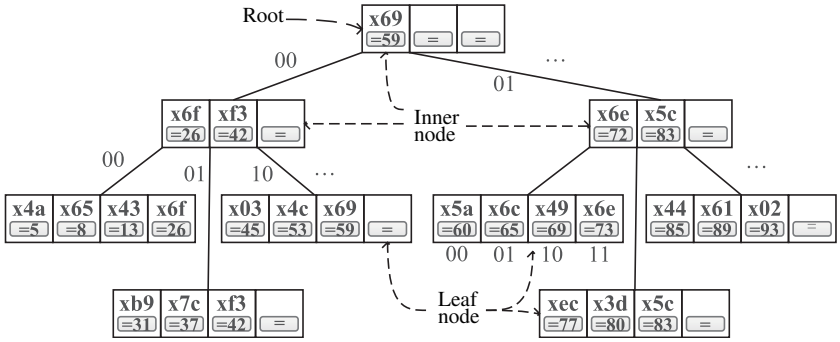

Figure 1 Code tree with three slots in inner nodes and four slots in leaves.

Algorithm 1 Encoding algorithm

Require: Root of code tree T, decryption key sk, encrypted element c;
Ensure: The code value o of c;
1: $o = \emptyset$, $p =$ root;
2: **for** p is not a leaf **do**
3: Decrypt the inner node p with sk and locate the child q to access;
4: Embed the ordinal of q into o;
5: $p = q$;
6: **end for**
7: Decrypt the leaf node and locate the element e that equals to $x = \mathrm{Dec}(c)$;
8: **if** No element is located **then**
9: Return $\perp$;
10: **end if**
11: Embed the ordinal of e into o;
12: Return o;

randomized ciphertexts (e.g., RSA, IBE, HE, and FHE). The former leads to additional computation and the latter makes it intractable to process the calculation directly over ciphertexts by introducing a homomorphic encryption.

We present a variant of B^+-tree to determine the ciphertext's code, called code tree, in which each code generated by tree traversal has the same length. Hence, it avoids postfix extension. Furthermore, our code tree provides more stable accessing efficiency than that presented in [3]. Then, two newly-defined lookup tables are designed for transforming randomized ciphertexts into order preserving codes.

The code tree is maintained in the same manner as a pure B^+-tree with two significant differences. In a code tree, the doubly linked list in the leaves is removed, and additionally, the number of slots in the inner nodes is only one less than that in the leaves. In general, the elements in the left hand child trees are strictly smaller than those in the right hand child trees. The elements in the nodes and leaves are encrypted by a homomorphic encryption algorithm and are not accessible without a correct decryption key. A code tree T is illustrated in Figure 1. The elements in plaintext (decimal value) are invisible and the elements in ciphertext (hexadecimal value) can be accessed only by a correct decryption key.

In general, a code tree has s slots in a leaf and $s-1$ slots in an inner node, where $s = 2^\ell$ and $2 \leqslant \ell \in \mathbb{N}^*$. To transform the original plaintext, the child trees from leftmost to rightmost in inner nodes are assigned ordinal $\{0, 1, \ldots, 2^\ell - 1\}$. Similarly, the elements from leftmost to rightmost in leaves are assigned ordinal $\{0, 1, \ldots, 2^\ell - 1\}$. In order to encode an element, code tree T is recursively accessed and all ordinals of each touched node are concatenated into an empty-initialized code. The process of encoding is formally defined in Algorithm 1. The algorithm starts from initialization (Line 2). Code tree T is recursively accessed by a loop and the ordinals of each inner node are embedded into the code (Lines 3–7). If no element is located, the algorithm is terminated (Lines 8–10). Finally, the code is obtained by embedding the ordinal of the located element and is returned (Lines 11, 12).

With the code tree and Algorithm 1, an arbitrary element in plaintext can be transformed into an order preserving code. For example, as shown in Figure 1, element 69 is transformed to $18 = \{010010\}$.

Table 1 Partial addition preserving lookup (APL) table

Plaintext	Ciphertext	Trapdoor	Order preserving code
...	...	...	...(=...)
26	x6f	26P	000011(=3)
...	...	...	...(=...)
59	x69	59P	001010(=10)
...	...	...	...(=...)
85	x44	85P	011000(=24)
...	...	...	...(=...)

Table 2 Partial product preserving lookup (PPL) table

Plaintext	Ciphertext	Trapdoor No. 1	Trapdoor No. 2	Order preserving code
5	x4a	5P	$\hat{e}(P,P)^5$	000000(=0)
8	x65	8P	$\hat{e}(P,P)^8$	000001(=1)
13	x43	13P	$\hat{e}(P,P)^{13}$	000010(=2)
...	...	...	...	...(=...)
65	x6c	65P	$\hat{e}(P,P)^{65}$	010001(=17)
...	...	...	...	...(=...)

Such codes obviously preserve the ordering of the original elements.

In order to repetitively utilize the mapping between ciphertexts and order preserving codes, a specific lookup table is maintained continuously as suggested in [3, 21, 25]. As stated previously, such a lookup table is not sufficient to provide hOPE with abilities of homomorphic operations. For example, given a newly arrived query of q, nobody can look up the corresponding code, since q's new ciphertext is not equal to that existing in the lookup table. Furthermore, not every homomorphic encryption supports consistent homomorphic operations. For instance, Paillier's method [12] supports only homomorphic addition and FHE [13–15] supports homomorphic product additionally. Hence, we redesigned the lookup tables and propose different kinds of tables for different homomorphic operations.

Definition 8 (Addition preserving lookup (APL) table)**.** An APL table is empty-initialized. Each item item $= \langle c, G, o\rangle$ is a triple tuple, where c is a ciphertext under HE/FHE, G is an element belonging to $\mathbb{G}$, and o is a code for preserving the ordering of plaintexts. $\mathbb{G}$ is an additive cyclic group with order $q \in \mathbb{Z}^*$. Both $\mathbb{G}$ and an HE/FHE scheme are selected according to a security parameter κ.

In the APL table, G is a trapdoor for transforming randomized ciphertexts into order preserving codes and P is a generator of $\mathbb{G}$. For a given plaintext m and the corresponding ciphertext c, the order preserving code o is generated by Algorithm 1. The trapdoor is $G = mP$. Then, the generated triple tuple is added to the APL table. A partial instance of the APL table for the code tree in Figure 1 is shown in Table 1. In fact, the plaintexts are listed here to facilitate understanding. Indeed, they are not visible for the table holder. The same applies to the plaintexts in Table 2.

Obviously, the trapdoor for a plaintext is deterministic and supports homomorphic addition. For arbitrary plaintexts x and y, $xP + yP = (x+y)P \in \mathbb{G}$. Indeed, nobody can derive x, y or $x+y$ from xP, yP, or $(x+y)P$, because of the intractability of the ECDL problem. Thus far, the code for plaintext x can be derived by matching the trapdoors in the APL table. However, the homomorphic product cannot be supported, since xyP cannot be derived from xP and yP according to the intractability of the CDH problem. In practice, x, y, $x+y$, and xy must be sufficiently large to guarantee the intractability of the ECDL and CDH problems. This limitation is resolved in Subsection 5.1, since x may be small in practice.

Definition 9 (Product preserving lookup (PPL) table)**.** The PPL table is based on the APL table, and each item in the PPL table has an additional element H, which is an element belonging to an multiplicative cyclic group $\mathbb{H}$. Hence, the item is represented as item $= \langle c, G, H, o\rangle$. A bilinear pairing map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$ and an HE/FHE scheme are selected according to a security parameter κ. Both $\mathbb{G}$ and $\mathbb{H}$ have the same order $q \in \mathbb{Z}^*$.

In the PPL table, both the trapdoors can be used for transforming randomized ciphertexts into order

preserving codes. For a given plaintext m, $H = \hat{e}(P,P)^m$. Other elements are generated in the same manner, as shown in the APL table. Then, the generated quadruple tuple is added to the PPL table. Specifically, a partial instance of the PPL table for the code tree in Figure 1 is shown in Table 2.

By introducing the second trapdoor, homomorphic products are supported, since for arbitrary plaintexts x and y, $\hat{e}(xP, yP) = \hat{e}(P,P)^{xy}$. However, this homomorphic product can be performed only once, since there is no efficient algorithm to compute $\hat{e}(P,P)^{xyz}$ only from xP, yP, and zP because of the intractability of the CBDH problem. For continuous homomorphic products, after a homomorphic product, plaintext xy is first derived from the corresponding ciphertext by decrypting with sk and the first trapdoor xyP can be set. Then, $\hat{e}(P,P)^{xyz}$ can be derived from $\hat{e}(xyP, zP)$. It remains to avoid divulging sk and this is discussed in Subsection 5.1. Additionally, x, y, and z must be sufficiently large to guarantee the intractability of the CBDH problem. This limitation is also resolved in Subsection 5.1.

Notably, it is extremely expensive to generate an order preserving code of the homomorphic result, because of the recursive traversal of the code tree, in which the participation of a third-party or interaction is inevitable, as stated in Subsection 5.1. By introducing the APL or PPL table, this repeated process of generating order preserving code for the homomorphic result is avoided. For example, at a certain stage of the protocol, a ciphertext $c = c_1 + c_2$ is derived and the corresponding order preserving code o is calculated by Algorithm 2. There is no doubt that c will exists in the APL or PPL table after this process. Furthermore, if $c = c_3 + c_4$ is derived, Algorithm 2 is not executed, since the corresponding order preserving code o can be derived by directly looking up the APL or PPL table. Hence, both the APL and PPL tables reduce the computation overhead in practice.

3 hOPE with the ability of homomorphic addition

This section presents a concrete hOPE scheme with the ability of homomorphic addition, namely AhOPE.

AhOPE.Setup(κ). Take as input security parameter κ. Pick up an HE/FHE scheme Enc (the corresponding decryption algorithm is Dec) satisfying at least IND-OCPA secure, and randomly select an additive cyclic group $\mathbb{G}$ with order $q \in \mathbb{Z}^*$ and a generator $P \in \mathbb{G}$. The system parameter is $\mathrm{SP} = (\mathrm{Enc}, \mathrm{Dec}, P, \mathbb{G})$.

On the one hand, an empty-initialized APL table $\mathbb{L}$ is maintained for mapping ciphertexts into order preserving codes. The item in $\mathbb{L}$ is represented as $L = \langle c, G, o\rangle$, which is the complete ciphertext. On the other hand, an empty-initialized code-tree T is maintained for adding a new item to APL table $\mathbb{L}$. The key value is the randomized ciphertext c under the HE/FHE scheme, as shown in Figure 1.

AhOPE.KeyGen(SP). Take as input SP, and generate an encryption (public) key pk and a decryption (private) key sk for Enc and Dec, respectively.

AhOPE.Enc(SP, pk, m). Take as input SP, pk, and a plaintext m. Ciphertext L is obtained as follows.

(1) Compute $c = \mathrm{Enc}(m)$ and $G = mP \in \mathbb{G}$.

(2) Look up the corresponding item $L_i = \langle c_i, G_i, o_i\rangle$ in APL table $\mathbb{L}$ by matching G. If the corresponding item is found, skip to Step (3); otherwise, skip to Step (4).

(3) The ciphertext is $L = L_i = \langle c_i, G_i, o_i\rangle$, where $G_i = G$. Then, the process is terminated.

(4) The value of o is determined by the code location algorithm as shown in Algorithm 2.

(5) Insert $L = \langle c, G, o\rangle$ to APL table $\mathbb{L}$, and c to code tree T. Then, code tree T and APL table $\mathbb{L}$ are updated. Finally, Step (2) is re-executed.

Algorithm 2 describes the code location algorithm in detail. The process starts from accessing root (Line 1). A loop is performed to find the location at which to insert c (Lines 2–6). Then, the homomorphic ciphertext c is added to code tree T (Lines 7–11). Because of the change in code tree T, the codes in the APL table are updated (Line 12). Using this algorithm, the newly-arrived element x exists in T. Finally, Step (2) in AhOPE.Enc can lookup the corresponding code and the complete ciphertext can be derived.

It is noteworthy that, in Lines 3 and 6 in Algorithm 2, decryption key sk is an inevitable for decrypting the nodes in T. However, the decryption key sk can be held only by a trusted party who is not always

Algorithm 2 Code location algorithm

Require: Code tree T, APL table $\mathbb{L}$, encrypted element c, decryption key sk and trapdoor G.
1: $p = \text{root}$;
2: **for** p is not a leaf **do**
3: Decrypt inner node p with decryption key sk and locate the child q to access;
4: $p = q$;
5: **end for**
6: Decrypt leaf p and locate $e = \text{key}_i$ such that $\text{key}_i \leqslant c < \text{key}_{i+1}$ where key is the array including all keys in leaf;
7: **if** Element is located **then**
8: Add c into the current location as a standard B$^+$-tree does;
9: **else**
10: Add c into the rightest location in B$^+$-tree;
11: **end if**
12: Update all the codes changed in APL table $\mathbb{L}$ according to Algorithm 1;

under the management of T and $\mathbb{L}$. This concern is eliminated latter in Subsection 5.1.

AhOPE.Dec(SP, sk, c). Take as input SP, sk, and a homomorphic ciphertext c. Output $m = \text{Dec}(c)$.

AhOPE.Addition(SP, L_1, L_2). Take as input SP, and two ciphertexts $L_1 = \langle c_1, G_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, o_2\rangle$ for m_1 and m_2, respectively. The sum L is generated as follows.

(1) Compute $c = c_1 + c_2$ and $G = G_1 + G_2$.

(2) Look up the corresponding item $L_i = \langle c_i, G_i, o_i\rangle$ in APL table $\mathbb{L}$ by matching G. If the corresponding item is found, skip to Step (3); otherwise, skip to Step (4).

(3) The ciphertext is $L = L_i = \langle c_i, G_i, o_i\rangle$, where $G_i = G$. Then, the process is terminated.

(4) The value of o is determined by Algorithm 2 with code-tree T.

(5) Insert $L = \langle c, G, o\rangle$ to the APL table $\mathbb{L}$, and c to the code tree T. Then, code tree T and APL table $\mathbb{L}$ are updated. Finally, Step (2) is re-executed.

AhOPE.Comp(SP, L_1, L_2). Take as input SP, and two ciphertexts $L_1 = \langle c_1, G_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, o_2\rangle$. Outputs the comparison result between o_1 and o_2.

Obviously, the algorithm outputs "$o_1 < o_2$" if $m_1 = \text{Dec}(c_1) < m_2 = \text{Dec}(c_2)$. "$o_1 = o_2$" is the result, when $m_1 = \text{Dec}(c_1) = m_2 = \text{Dec}(c_2)$; otherwise, the algorithm outputs "$o_1 > o_2$".

3.1 Completeness

Theorem 2 (Completeness of AhOPE)**.** The proposed AhOPE algorithm is complete.

Proof. The proof consists of the following three parts.

Proof of decryption ability. For an arbitrary given ciphertext $L = \langle c, G, o\rangle$, the plaintext m can be derived by $\text{Dec}(c)$, since $c = \text{Enc}(m)$ under the adopted HE/FHE scheme.

Proof of additivity. For arbitrary given ciphertexts $L_1 = \langle c_1, G_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, o_2\rangle$ for plaintexts m_1 and m_2, respectively, the sum $L = \langle c, G, o\rangle$ of L_1 and L_2 is calculated by AhOPE.Addition. Here, $c = c_1 + c_2$ and $G = G_1 + G_2$. o is generated by inserting c to code tree T and transforming c according to Algorithms 2 and 1, respectively. Obviously, $m_1 + m_2 = \text{Dec}(c) = \text{Dec}(c_1 + c_2)$, since the adopted HE/FHE algorithm owns the ability of homomorphic addition. Meanwhile, $G = G_1 + G_2 = m_1P + m_2P = (m_1 + m_2)P$ since G is an additive cyclic group. Hence, additivity is satisfied.

Proof of comparability. On the one hand, arbitrary ciphertexts $L_1 < L_2$, where $L_1 = \langle c_1, G_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, o_2\rangle$, are generated only by either AhOPE.Enc or AhOPE.Addition. It is obvious that $o_1 < o_2$. Both c_1 and c_2 do exist in code tree T. Therefore, c_1 must lie on the left of c_2. Consequently, the corresponding m_1 must lie on the left of m_2. Hence, $m_1 < m_2$ is satisfied.

On the other hand, for arbitrary plaintexts $m_1 < m_2$, the ciphertexts $L_1 = \langle c_1, G_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, o_2\rangle$ are generated only by AhOPE.Enc. Both c_1 and c_2 are inserted to code tree T. c_1 must lie on the left of c_2, since $m_1 < m_2$. Consequently, $o_1 < o_2$, and thus, $L_1 < L_2$ is satisfied.

Hence, for arbitrary ciphertexts L_1 and L_2, $L_1 < L_2$ iff $m_1 < m_2$. Comparability is satisfied.

3.2 Security

Lemma 1. $\Pr[\mathbb{E}_1] \leqslant \epsilon_1$, where ϵ_1 is the maximum advantage for $\mathcal{A}$ breaking a random oracle $\mathcal{O}$, and $\mathbb{E}_1$ is the event that $\mathcal{A}$ distinguishes scalar multiplication in an additive cyclic group from $\mathcal{O}$.

Proof. Let hash be the map from an integer a to a scalar multiplication value $aP \in \mathbb{G}$. We first demonstrate that hash is a pseudo-random map. Accordingly, the advantage for $\mathcal{A}$ distinguishing hash from a random oracle $\mathcal{O}$ is proved to be negligible.

First part. Assume that hash is not a pseudo-random map. There exists at least a probabilistic polynomial-time distinguisher who distinguishes the differences between $xP \in \mathbb{G}$ and $yP \in \mathbb{G}$, where x and y are randomly selected and P is the generator of $\mathbb{G}$. We can construct a simulator to derive x from $xP \in \mathbb{G}$. Given x and y, the challenger picks up a random bit $b \in \{0, 1\}$ with $\Pr[b = 0] = 1/2$. If $b = 0$, the challenger returns xP; otherwise, he returns yP. The simulator can easily guess a correct $b' = b$, since he can distinguish the two elements in $\mathbb{G}$. Obviously, the simulator resolves the ECDL problem, which is inconsistent with the fact. Hence, hash is a pseudo-random map.

Second part. Adversary $\mathcal{A}$ submits $\mathbb{X}$ and $\mathbb{Y}$ with the restriction that $\hat{v}(\mathbb{X}) = \hat{v}(\mathbb{Y})$. Then, the simulator picks up a random bit $b \in \{0, 1\}$ with a probability such that $\Pr[b = 0] = 1/2$. If $b = 0$, the simulator answers the adversary $\mathcal{A}$ with $\mathbb{X} \cdot P$[4]; otherwise, he returns $\mathbb{Y} \cdot P$. Adversary $\mathcal{A}$ guesses a bit b' and wins the game when $b = b'$. In such a case, the event $\mathbb{E}_1$ occurs.

We proceed inductively in the number of elements in $\mathbb{X}$ and $\mathbb{Y}$ to be challenged. The base case is when the size of both $\mathbb{X}$ and $\mathbb{Y}$ is 1, and obviously, the adversary wins the game with advantage $\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},1} = \epsilon_1$, since the code of both $x \in \mathbb{X}$ and $y \in \mathbb{Y}$ is 0. Now consider that after k times of encryptions, adversary $\mathcal{A}$ has k order preserving codes for the first k elements in both $\mathbb{X}$ and $\mathbb{Y}$, and $\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k} = \tau(\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k-1})$ remains a negligible function. There are two possibilities in the $k + 1$ round of encryptions.

Note that, since both $\mathbb{X}$ and $\mathbb{Y}$ have the same order relations, both x_{i+1} and y_{i+1} do or do not exist in the APL table. The first possibility is that both x_{i+1} and y_{i+1} are already in the APL tables. Obviously, adversary $\mathcal{A}$ gains nothing more valuable than he gains in the k round of encryption. Hence, the advantage for $\mathcal{A}$ winning the game is $\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k+1} = \mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k}$. The second possibility is that neither x_{i+1} nor y_{i+1} is already in the APL tables. According to code trees T_1 and T_2 and Algorithm 2, the path to recursively locate the code must be the same. Only the completely consistent paths are divulged to adversary $\mathcal{A}$. No further available information for distinguishing the differences between $\mathbb{X} \cdot P$ and $\mathbb{Y} \cdot P$ is leaked. Hence, the advantage for $\mathcal{A}$ winning the game remains $\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k+1} = \mathrm{Adv}^{\mathcal{O}}_{\mathcal{A},k}$.

Finally, $\Pr[\mathbb{E}_1] \leqslant \epsilon_1$ by recurse, where ϵ_1 is the advantage for $\mathcal{A}$ breaking a random oracle.

In fact, the proof is interactive. We omit the interactive part here because this part can be well resolved by avoiding leakage of the secret key in Algorithm 2. As shown in Subsection 5.1, the cloud can achieve Algorithm 2 by employing interactions with an additional trusted party who owns the secret key.

Theorem 3 (Security of AhOPE)**.** The proposed AhOPE algorithm is IND-O^{2}CPA secure, iff the adopted HE/FHE is IND-OCPA secure at least and ECDL problem is intractable in probabilistic polynomial time. The advantage for an adversary breaking such an AhOPE is

$$\mathrm{Adv}^{\mathrm{AhOPE}}_{\mathcal{A}} \leqslant \epsilon_1 + \epsilon_2,$$

where ϵ_1 and ϵ_2 are the maximum advantages for an adversary breaking a random oracle and the adopted HE/FHE.

Proof. Let $\mathcal{A}$ be any adversary restricted in probabilistic polynomial time. We present an ideal model in which the scalar multiplication aP in the additive cyclic group $\mathbb{G}$ is replaced by a random oracle $\mathcal{O}$. In ideal model, the proposed AhOPE is IND-O^{2}CPA secure, since no adversary can break the random oracles. The concrete model strictly follows the concrete AhOPE. If $\mathcal{A}$ cannot distinguish the differences between the ideal and concrete model, the proposed AhOPE is IND-O^{2}CPA secure.

Ideal model. The random oracle $\mathcal{O}$ maintains a list $\mathbb{T} = \langle m, G \rangle$ mapping plaintext to a deterministic random value. The advantage $\mathrm{Adv}^{\mathcal{O}}_{\mathcal{A}}$ for $\mathcal{A}$ breaking $\mathcal{O}$ is ϵ_1 at most. The advantage $\mathrm{Adv}^{\mathrm{HE}}_{\mathcal{A}}$ for $\mathcal{A}$

4) Here, $\mathbb{X} \cdot P$ means $\{x_i P | x_i \in \mathbb{X}\}$. In addition, $\mathbb{Y} \cdot P$ is defined in the same manner.

breaking the adopted HE/FHE algorithm is ϵ_2 at most. Both ϵ_1 and ϵ_2 are negligible functions.

Concrete model. Such a model strictly follows the proposed AhOPE.

Both $\mathbb{X}$ and $\mathbb{Y}$ are challenged with the restriction that $\hat{v}(\mathbb{X}) = \hat{v}(\mathbb{Y})$. Adversary $\mathcal{A}$'s breaking of AhOPE is furthermore divided into two independent events.

$\mathbb{E}_1$: $\mathcal{A}$ distinguishes the scalar multiplication in the additive cyclic group from the random oracle $\mathcal{O}$.

$\mathbb{E}_2$: $\mathcal{A}$ distinguishes the differences between two sequences $E(\mathbb{X})$ and $E(\mathbb{Y})$ of homomorphic ciphertexts.

Hence, the advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{AhOPE}}$ for $\mathcal{A}$ breaking the AhOPE is also the addition $\Pr[\mathbb{E}_1] + \Pr[\mathbb{E}_2]$ at most. We have proved $\Pr[\mathbb{E}_1] \leqslant \epsilon_1$ in Lemma 1, where ϵ_1 is negligible function.

If event $\mathbb{E}_2$ occurs with a probability greater than ϵ_2, it is obvious that a simulator can be constructed for an adversary having advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{HE}} > \epsilon_2$ to break the adopted HE/FHE. This is inconsistent with the fact. Hence, $\Pr[\mathbb{E}_2] \leqslant \epsilon_2$.

Briefly, the advantage $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{AhOPE}} \leqslant \epsilon_1 + \epsilon_2$ for $\mathcal{A}$ breaking the AhOPE is negligible.

4 hOPE with the ability of homomorphic product

Now, a concrete hOPE is presented with the ability of homomorphic product, called PhOPE.

PhOPE.Setup(κ). Take as input a security parameter κ. Pick up an FHE scheme Enc (The corresponding decryption algorithm is Dec) satisfying at least IND-OCPA secure, randomly select a bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{H}$ with the order $q \in \mathbb{Z}^*$ for both $\mathbb{G}$ and $\mathbb{H}$, and choose a generator $P \in \mathbb{G}$. The system parameter is SP=(Enc, Dec, $P, \hat{e}$).

On the one hand, according to Definition 9, an empty-initialized PPL table $\mathbb{L}$ is maintained for mapping ciphertexts into order preserving codes. The item in $\mathbb{L}$ is the ciphertext represented as $L = \langle c, G, H, o\rangle$. On the other hand, code tree T is maintained in the same manner as in AhOPE.

PhOPE.KeyGen(SP). The process is the same as that of AhOPE.KeyGen.

PhOPE.Enc(SP, pk, m). Take as input SP, pk, and a plaintext m. The ciphertext L is obtained as follows.

(1) Compute $c = \mathrm{Enc}(m)$, $G = mP \in \mathbb{G}$, and $H = \hat{e}(P, P)^m \in \mathbb{H}$.

(2) Look up the corresponding item $L_i = \langle c_i, G_i, H_i, o_i\rangle$ in PPL table $\mathbb{L}$ by matching G or H. If the corresponding item is found, skip to Step (3); otherwise, skip to Step (4).

(3) The ciphertext is $L = L_i = \langle c_i, G_i, H_i, o_i\rangle$, where $G_i = G$ or $H_i = H$. Then, the process is terminated.

(4) o is determined by the code location algorithm with code-tree T as shown in Algorithm 2.

(5) Insert $L = \langle c, G, H, o\rangle$ to APL table $\mathbb{L}$, and insert c to code tree T. Then, code tree T and PPL table $\mathbb{L}$ are updated. Finally, Step (2) is re-executed.

PhOPE.Dec(SP, sk, c). The process is the same as that of AhOPE.Dec.

PhOPE.Addition(SP, L_1, L_2). Take as input SP, and two ciphertexts $L_1 = \langle c_1, G_1, H_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, H_2, o_2\rangle$. The sum L is generated as follows.

(1) Compute $c = c_1 + c_2$ and $G = G_1 + G_2$.

(2) Look up the corresponding item $L_i = \langle c_i, G_i, H_i, o_i\rangle$ in PPL table $\mathbb{L}$ by matching G or H. If the corresponding item is found, skip to Step (3); otherwise, skip to Step (4).

(3) The ciphertext is $L = L_i = \langle c_i, G_i, H_i, o_i\rangle$, where $G_i = G$ or $H_i = H$. Then, the process is terminated.

(4) The value of o is determined by Algorithm 2 with code-tree T.

(5) Compute $H = \hat{e}(G, P)$.

(6) Insert $L = \langle c, G, H, o\rangle$ to PPL table $\mathbb{L}$, and insert c to code tree T. Then, code tree T and PPL table $\mathbb{L}$ are updated. Finally, Step (2) is re-executed.

PhOPE.Product(SP, L_1, L_2). Take as input SP, and two ciphertexts $L_1 = \langle c_1, G_1, H_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, H_2, o_2\rangle$. The product L is generated as follows.

(1) Compute $c = c_1 \times c_2$, and $H = \hat{e}(G_1, G_2)$.

(2) Look up the corresponding item $L_i = \langle c_i, G_i, H_i, o_i\rangle$ in PPL table $\mathbb{L}$ by matching H. If the corresponding item is found, skip to Step (3); otherwise, skip to Step (4).

(3) The ciphertext is $L = L_i = \langle c_i, G_i, H_i, o_i\rangle$, where $G_i = G$ or $H_i = H$. Then, the process is terminated.

(4) The value of o is determined by Algorithm 2 with the code-tree T.

(5) Compute $m = \mathrm{Dec}(c)$, and $G = mP$.

(6) Insert $L = \langle c, G, H, o\rangle$ to PPL table $\mathbb{L}$, and insert c to code tree T. Then, code tree T and PPL table $\mathbb{L}$ are updated. Finally, Step (2) is re-executed.

PhOPE.Comp(SP, L_1, L_2). The process is the same as that of AhOPE.Comp.

The risk of divulging decryption key sk is discussed in detail in Subsection 5.1.

4.1 Completeness

Theorem 4 (Completeness of PhOPE)**.** The proposed PhOPE scheme is complete.

Proof. The decryption ability, additivity and comparability can be proved by generalizing the conclusion in Theorem 2.

Proof of productivity. For arbitrary given ciphertexts $L_1 = \langle c_1, G_1, H_1, o_1\rangle$ and $L_2 = \langle c_2, G_2, H_2, o_2\rangle$, the product $L = \langle c, G, H, o\rangle$ of L_1 and L_2 is calculated by PhOPE.Product. Here, $c = c_1 + c_2$, $G = mP = (m_1 \times m_2)P$ by Step (5) in **PhOPE.Product**, and $H = \hat{e}(G_1, G_2) = \hat{e}(P, P)^{m_1 \times m_2}$ by Step (1) in **PhOPE.Product**. o is generated by inserting c to code tree T and encoding c according to Algorithms 2 and 1, respectively. Obviously, $m_1 \times m_2 = \mathrm{Dec}(c) = \mathrm{Dec}(c_1 \times c_2)$ since the adopted FHE algorithm owns the ability of homomorphic product. Hence, productivity is satisfied.

In a word, PhOPE is complete.

4.2 Security

Theorem 5 (Security of PhOPE)**.** The proposed PhOPE scheme is IND-O^{2}CPA secure, iff the adopted FHE is IND-OCPA secure at least, the ECDL problem in both $\mathbb{G}$ and $\mathbb{H}$ are intractable in probabilistic polynomial time, and the BPI problem of the bilinear pairing map $\hat{e}$ is also intractable in probabilistic polynomial time. The advantage for an adversary breaking such a PhOPE is defined as

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PhOPE}} \leqslant 2\epsilon_1 + \epsilon_2,$$

where ϵ_1 and ϵ_2 are the maximum advantages for an adversary breaking a random oracle and the adopted FHE, respectively.

Proof. $\mathcal{A}$ is the adversary in probabilistic polynomial time. The ideal model and concrete model are both adopted, in which the power multiplication $\hat{e}(P, P)^a$ in the multiplicative cyclic group $\mathbb{H}$ is replaced by another random oracle $\mathcal{O}'$. The concrete model follows the proposed PhOPE strictly.

Adversary $\mathcal{A}$'s breaking of the ideal model is furthermore divided into three events.

$\mathbb{E}_1$: $\mathcal{A}$ distinguishes the scalar multiplication in the additive cyclic group from a random oracle $\mathcal{O}$.

$\mathbb{E}_2$: $\mathcal{A}$ distinguishes the power multiplication in the multiplicative cyclic group from another random oracle $\mathcal{O}'$.

$\mathbb{E}_3$: $\mathcal{A}$ distinguishes the differences between two sequences $E(\mathbb{X})$ and $E(\mathbb{Y})$ of homomorphic ciphertexts.

Here, $\mathbb{E}_3$ is independent of $\mathbb{E}_1$ and $\mathbb{E}_2$ obviously. It is proved that $\mathbb{E}_1$ is independent of $\mathbb{E}_2$. The event $\mathbb{E}_1$ occurs iff adversary $\mathcal{A}$ resolves the ECDL problem in $\mathbb{G}$, and also $\mathbb{E}_2$ occurs iff adversary $\mathcal{A}$ resolves the ECDL problem in $\mathbb{H}$. In order to prove the independence of $\mathbb{E}_1$ and $\mathbb{E}_2$, we prove that the ECDL problem in $\mathbb{H}$ is intractable, regardless of whether the ECDL problem in $\mathbb{G}$ is intractable.

Suppose that the ECDL problem in $\mathbb{H}$ is not intractable. On the one hand, the ECDL problem in $\mathbb{G}$ is intractable. Given $Q = xP$ and $\hat{e}(Q, P) = \hat{e}(P, P)^x$, x is easily derived by logarithm in $\mathbb{H}$. Therefore, we can construct a simulator to resolve the ECDL problem in $\mathbb{G}$, which is inconsistent with the precondition. On the other hand, the ECDL problem in $\mathbb{G}$ is not intractable. Given a known $Q = xP$, a known $\hat{e}(Q, R)$, and an unknown $R = yP$, x is easily derived by the logarithm in $\mathbb{H}$, and xy is easily derived by

the logarithm in $\mathbb{H}$. We can compute y by dividing xy by x, and furthermore, compute $R = yP$. It is still inconsistent with the intractability of the BPI problem. Hence, the ECDL problem in $\mathbb{H}$ is intractable.

For a similar reason, the ECDL problem in $\mathbb{G}$ is intractable, regardless of whether ECDL problem in $\mathbb{H}$ is intractable. In summary, all events are independent of each other.

Obviously, the advantage for $\mathcal{A}$ breaking PhOPE is $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PhOPE}} \leqslant \sum_{i=1}^{3} \Pr[\mathbb{E}_i]$. By simply generalizing the conclusion in Theorem 3, we have $\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PhOPE}} \leqslant 2\epsilon_1 + \epsilon_2$.

5 Deployment and comparison

In this section, we discuss and resolve several issues of concern with respect to the deployment, through which the proposed hOPE schemes are proved to be indeed practical. We focus on shifting the heavy burden of maintaining code tree T and lookup table $\mathbb{L}$ while resolving the risk of divulging the decryption key. Additionally, we remove the vulnerability of hard problems (i.e., ECDLP, CDH, CBDH, etc.) when plaintexts have a quite small numeric value (in particular, smaller than 64 bits). Then, we exhaustively examine and analyze both hOPEs in depth in comparison with state-of-art related methods.

5.1 Deployment

Subcontracting. In both AhOPE and PhOPE, code tree T, the APL table, and the PPL table (i.e., the expensive components) are maintained by the client who owns the decryption key. However, this always imposes a very heavy cost on the client, who has limited in-house computation and storage resources. Because of this, the most expensive components are subcontracted to a third party [16, 26, 27]. All the expensive components can be subcontracted to either a trusted third party (TTP) with all the secret keys or an untrusted third party (UTP) without any secret keys. In the case of using a TTP [26, 27], which can substitute a client to submit a query to a remote service provider, the client undertakes the minimum burden during querying[5)]. During each usage of the secret keys in the case of using a UTP [16], the client coordinate with the UTP to achieve the following interaction to avoid the leakage of secret keys.

Implementation of interactions. In both schemes, the interactions occur at Steps 4 and 8 of Algorithm 1, Steps 3 and 6 of Algorithm 2, and Step (5) in PhOPE.Product while waiting for the removal of the security concerns, since the decryption key cannot be revealed to an untrusted party (i.e., untrusted proxy and server). Step 4 of Algorithm 1 and Step 3 of Algorithm 2 are executed as follows. The server sends an encrypted inner node to the proxy (resp. client) for architecture with the TTP (resp. UTP). When it receives the encrypted inner node, the proxy (resp. client) decrypts it and decides which child tree to access, and finally the proxy (resp. client) tells the server the chosen path. This interaction can be achieved through a public channel, since the information leaked will infer nothing about the plaintext, except the order. Similarly, Step 8 of Algorithm 1 and Step 6 of Algorithm 2 are executed, starting with the server sending the encrypted child node to the proxy (resp. client). The proxy (resp. client) decrypts it and locates the element equal to the query for Algorithm 1 and the minimal element larger than or equal to the query for Algorithm 2. Finally, the location is returned to the server. Step (5) in PhOPE.Product is executed by a single interaction between the proxy (resp. client) and the server. The sever sends the homomorphic ciphertext c to the proxy (resp. client). Then, he decrypts it, computes $G = mP$, and returns it to the server. In a word, the decryption key is held only by a trusted party (trusted proxy and client) and such interactions exert no negative effect on the security of hOPEs.

Handling the vulnerability of hard problems. Both AhOPE and PhOPE rely on the intractability of the several hard problems listed in Subsection 2.2, in which ECDL is the fundamental one for the others. In some specific practical applications, x is small and ECDL is easily solved. To overcome this

5) Introducing a third-party with the secret key is indeed a potential privacy risk. However, in order to achieve IND-OCPA secure, it has been proved in [3] that ciphertexts in OPE must be mutable, and the current constructions need to employ a third-party with the secret key to achieve mutability of ciphertexts [3, 21, 25]. How to avoid using a third-party with the secret key is beyond the scope of this paper and is still an open problem.

Table 3 Theoretical comparisons

Scheme	Security	Element comparison	Metric computation	Metric comparison
Ref. [3]	IND-OCPA	Yes	No	No
Ref. [21]	IND-OCPA	Yes	No	No
Ref. [25]	IND-OCPA	Yes	No	No
Ref. [12]	IND-CPA	No	Yes	No
Ref. [15]	IND-CPA	No	Yes	No
AhOPE	IND-O^{2}CPA	Yes	Yes	Yes
PhOPE	IND-O^{2}CPA	Yes	Yes	Yes
Scheme	Trapdoor length	Ciphertext length	Number of interactions	Code length
Ref. [3]	–	Short	$\log_{2^{\ell-1}}(\frac{n+1}{2})$	$\ell \cdot \log_{2^{\ell-1}}(\frac{n+1}{2})$
Ref. [21]	–	Extremely Short	$\log_2(\frac{n+1}{2})$	$\log_2(\frac{n+1}{2})$
Ref. [25]	–	Extremely Short	$\log_2(\frac{n+1}{2})$	$\log_2(\frac{n+1}{2})$
Ref. [12]	–	Long	–	–
Ref. [15]	–	Long	–	–
AhOPE	len($\mathbb{G}$)	Long	$\log_{2^{\ell-1}}(n)$	$\ell \cdot \log_{2^{\ell-1}}(n)$
PhOPE	len($\mathbb{G}$) + len($\mathbb{H}$)	Long	$\log_{2^{\ell-1}}(n)$	$\ell \cdot \log_{2^{\ell-1}}(n)$

drawback, we introduce an additional secret key $\mathrm{sk}_\lambda = \{0,1\}^\lambda$, where λ is sufficiently long to guarantee the intractability of the ECDL problem. Assume plain space is $\mathbb{M} = \{0,1\}^n$.

In AhOPE, the order q of additive cyclic group $\mathbb{G}$ has $\lambda + n$ bit length at least. Hence, $|q| \geqslant \lambda + n$. When computing the trapdoor of plaintext m, $G = \mathrm{sk}_\lambda mP$ is calculated (Step (2) in AhOPE.Enc). However, when computing the homomorphic ciphertext, sk_λ does not still participate in the computation and $c = \mathrm{Enc}(m)$. In this manner, even when n is small, deriving $\mathrm{sk}_\lambda m$ from $\mathrm{sk}_\lambda mP$ is still intractable. Thus, m also cannot be derived. Furthermore, for plaintexts m, m_1 and m_2 such that $m = m_1 + m_2$, m's trapdoor is $G = (m_1 + m_2)\mathrm{sk}_\lambda P$, where $G_i = m_i \mathrm{sk}_\lambda P$. The functionality of the APL table is preserved.

In PhOPE, order q of $\mathbb{G}$ and $\mathbb{H}$ has $2(\lambda+n)$ bit length at least. Hence, $|q| \geqslant 2(\lambda+n)$. When computing the trapdoor of plaintext m, $G = \mathrm{sk}_\lambda mP$ is calculated and $H = \hat{e}(P,P)^{\mathrm{sk}_\lambda^2 m}$ is calculated (Step (2) in PhOPE.Enc). However, when computing the homomorphic ciphertext, sk_λ does not still participate in the computation and $c = \mathrm{Enc}(m)$. For given plaintexts m, m_1 and m_2 such that $m = m_1 + m_2$, the trapdoor for m is $H = \hat{e}(G_1, G_2) = \hat{e}(P,P)^{(\mathrm{sk}_\lambda^2 m_1 m_2)}$, where $G_i = \mathrm{sk}_\lambda m_i P$ and H is a valid trapdoor for m according to the definition of the PPL table. The functionality of the PPL table can also be preserved.

5.2 Comparison

Thus far, the concrete hOPEs have been formally defined. To the best of our knowledge, no previous method supports order preservation and homomorphic operation simultaneously. In this section, we compare AhOPE and PhOPE with other competitive proposed methods theoretically. We focus only on the mechanisms that supports provable security as ours does. In the comparison, we focus on security and efficiency in sequence, since the security issue is the most important side effect when outsourcing the encrypted data.

The theoretical comparison results are shown in Table 3. All the models in this paper and the referenced models [3, 12, 15, 21, 25] were proved to be semantic secure against the chosen-plaintext attack, which is considered to be a practical security guarantee to protect the user's sensitive data. The OPE methods [3, 21, 25] and homomorphic encryption algorithms [12, 15] offer either element comparison or metric computation. Both AhOPE and PhOPE offer both operations concurrently as well as the metric comparison. All the tree operations are performed directly over ciphertexts.

There is an additional trapdoor in AhOPE and PhOPE because of the order preservation of homomorphic ciphertexts. The increments of ciphertext compared with homomorphic ciphertext are len($\mathbb{G}$) and len($\mathbb{G}$) + len($\mathbb{H}$), respectively, which are not too expensive for practice. The negative effect of the number of interactions during the update of the code tree and lookup table is tremendous. Although the HE/FHE algorithms do not need any interactions, they do not satisfy the completeness of hOPE. In

counting the numbers, ℓ means that each node has 2^ℓ key values at most in B-tree as suggested in [3] and $2^{\ell-1}$ key values at least in our code tree. For fairness, we suppose that each node in B-tree also includes $2^{\ell-1}$ key values at least. Both the OPE scheme in [3] and ours need the least and identical interactions obtained by increasing ℓ to $\ell+1$ in our construction. Our code tree provides more stable interactions than the B-tree structure, because of the well-known characteristic of B^+-tree. In addition, the code length, as compared with homomorphic ciphertext and deterministic ciphertext, is negligible. In fact, the ciphertext length in [21] is the shortest, since in fact symmetric encryption is not employed. However, the method in [21] cannot support metric comparison over ciphertexts.

6 Related work

Element comparison, metric computation and comparison directly over ciphertexts constitute the core operations in secure data management. We investigated all the relevant approaches resolving the issues.

Ciphertext-based element comparison. OPE was first proposed in [1] and has recently been improved in terms of efficiency and security [2–5, 21, 25]. Boldyreva et al. [2] suggested an ideal security model, IND-OCPA, which was first realized by Popa et al. [3] in 2013. At the same time, other OPE schemes have been proved to be unable to satisfy IND-OCPA. Following the study in [3], Kerschbaum et al. [21] proposed an optimized OPE that provides a more stable code tree structure. OPEs based on a specific code tree are popular in the academic and industrial communities. Recently, Kerschbaum et al. [25] proposed a frequency-hiding OPE that avoids statistical attacks. This work is outside the scope of our research. However, not all the schemes can handle ciphertext-based metric computation and comparison. Choi et al. [28] resolves secure kNN query in 2-dimensional data by transmitting the metric computation to Voronoi cell enclosure evaluation. Such an approach is severely limited and cannot be generalized to arbitrary metric space.

Ciphertext-based metric computation. HE and FHE were the first type of approach for partially resolving such problem. HE was proposed by Paillier et al. [12] and FHE by Gentry et al. [13]. Later the studies [14, 15] focused on improving the efficiency of HE and FHE and achieved considerable progress. Elmehdwi et al. [16] introduced squared Euclidean distance to replace native Euclidean distance, based on HE. However, because of the construction of ciphertext, there is no efficient algorithm for comparing two ciphertexts directly. CryptDB [26] and Monomi [27] achieved the ciphertext-based metric computation by preprocessing such requirements. However, in practice, these requirements may be updated later and cannot be determined before subcontracting the encrypted database.

Ciphertext-based metric comparison. As stated in Section 1, simply combining OPE and HE/FHE cannot resolve this issue. The proposed method in [16] constituted the first attempts toward partially resolving this issue in a specific application scenario without provable security, in which only the encrypted results of metric computation need to be compared. For our application scenarios, the model in [16] is not appropriate, since all the ciphertexts in the database should be comparable.

In summary, there are few appropriate approaches that are well suited for our application scenarios.

7 Conclusion

Subcontracting encrypted data to a public cloud is a noteworthy means of providing both data confidentiality and data utility in the cloud computing era. hOPEs can handle element comparison, metric computation and metric comparison directly over ciphertext inherently, which constitute the issues that raise the most concern in the database community. Because of the security issue when considering operations and comparisons in ciphertext, an appropriate security model, IND-O^2CPA, was carefully designed. Then, the concrete hOPEs were proposed to resolve homomorphic addition and homomorphic product issues in addition to ciphertext comparisons, namely AhOPE and PhOPE, respectively.

We resolved urgent problem that arose during the practical deployment of hOPES. This proves that our proposed methods are practical and superior. We compared our models with three other state-of-the-art

OPE and HE/FHE solutions in terms of both security and efficiency. Our proposed models achieve an appropriate security level and provide sound efficiency. As part of our future work, we plan to investigate techniques for reducing the interactions during the update of the code tree and the lookup table.

Acknowledgements The work was supported by National Natural Science Foundation of China (Grant Nos. 61472298, 61672408, 61262073, 61472310, U1405255, 61662009), National High Technology Research and Development Program (Grant No. 2015AA016007), and China 111 Project (Grant No. B16037).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Agrawal R, Kiernan J, Srikant R, et al. Order preserving encryption for numeric data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (ICDE), Paris, 2004. 563–574
2 Boldyreva A, Chenette N, Lee Y, et al. Order-preserving symmetric encryption. In: Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques, Cologne, 2009. 224–241
3 Popa R A, Li F H, Zeldovich N. An ideal-security protocol for order-preserving encoding. In: Proceedings of IEEE Symposium on Security and Privacy (S&P), Berkeley, 2013. 463–477
4 Teranishi I, Yung M, Malkin T. Order-preserving encryption secure beyond one-wayness. In: Advances in Cryptology—ASIACRYPT. Berlin: Springer, 2014. 42–61
5 Mavroforakis C, Chenette N, O'Neill A, et al. Modular order-preserving encryption, revisited. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Melbourne, 2015. 763–777
6 Zhang H G, Han W B, Lai X J, et al. Survey on cyberspace security. Sci China Inf Sci, 2015, 58: 110101
7 Bentley J L. Multidimensional binary search trees used for associative searching. ACM Commun, 1975, 18: 509–517
8 Bayer R, McCreight E. Organization and maintenance of large ordered indexes. Acta Inform, 1972, 1: 173–189
9 Hartigan J A, Wong M A. Algorithm as 136: a k-means clustering algorithm. J Royal Stat Soc Ser C, 1979, 28: 100–108
10 Guttman A. R-trees: a dynamic index structure for spatial searching. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), Boston, 1984. 47–57
11 Rivest R L, Adleman L, Dertouzos M L. On data banks and privacy homomorphisms. In: Proceedings of Foundations of Secure Computation, Atlanta, 1978. 165–179
12 Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Advances in Cryptology—EUROCRYPT'99. Berlin: Springer, 1999. 223–238
13 Gentry C. Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC), Bethesda, 2009. 9: 169–178
14 Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. In: Advances in Cryptology—EUROCRYPT. Berlin: Springer, 2011. 129–148
15 Brakerski Z, Gentry C, Vaikuntanathan V. (Leveled) fully homomorphic encryption without bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, Cambridge, 2012. 309–325
16 Elmehdwi Y, Samanthula B, Jiang W. Secure k-nearest neighbor query over encrypted data in outsourced environments. In: Proceedings of IEEE 30th International Conference on Data Engineering (ICDE), Chicago, 2014. 664–675
17 Boneh D, Lynn B, Shacham H. Short signatures from the weil pairing. In: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. London: Springer, 2001. 514–532
18 Jiang Q, Wei F, Fu S, et al. Robust extended chaotic maps-based three-factor authentication scheme preserving biometric template privacy. Nonlinear Dynam, 2016, 83: 2085–2101
19 Koblitz N, Menezes A. Pairing-based cryptography at high security levels. In: Proceedings of IMA International Conference on Cryptography and Coding, Cirencester, 2005. 13–36
20 Jiang Q, Ma J, Li G, et al. Improvement of robust smart-card-based password authentication scheme. Int J Commun Syst, 2015, 28: 383–393
21 Kerschbaum F, Schroepfer A. Optimal average-complexity ideal-security order-preserving encryption. In: Proceedings

of ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, 2014. 275–286

22 Curtmola R, Garay J, Kamara S, et al. Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Secur, 2011, 19: 895–934

23 Kuzu M, Islam M S, Kantarcioglu M. Efficient similarity search over encrypted data. In: Proceedings of the IEEE 28th International Conference on Data Engineering (ICDE), Washington, 2012. 1156–1167

24 Demertzis I, Papadopoulos S, Papapetrou O, et al. Practical private range search revisited. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Francisco, 2016. 185–198

25 Kerschbaum F. Frequency-hiding order-preserving encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS), Denver, 2015. 656–667

26 Popa R A, Redfield C M S, Zeldovich N, et al. CryptDB: protecting confidentiality with encrypted query processing. In: Proceedings of the 23rd ACM Symposium on Operating Systems Principles, Cascais, 2011. 85–100

27 Tu S, Kaashoek M F, Madden S, et al. Processing analytical queries over encrypted data. Proc VLDB Endow, 2013, 6: 289–300

28 Choi S, Ghinita G, Lim H S, et al. Secure kNN query processing in untrusted cloud environments. IEEE TKDE, 2014, 26: 2818–2831