Parameter-Hiding Order-Revealing Encryption without Pairings

Cong Peng[1][0000−0002−9958−3255], Rongmao Chen[2]✉[0000−0002−5113−387X], Yi Wang[2][0000−0002−7456−2677], Debiao He[1]✉[0000−0002−2446−7436], and Xinyi Huang[3][0000−0003−0070−1707]

[1] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan, China
cpeng@whu.edu.cn, hedebiao@163.com

[2] School of Computer, National University of Defense Technology, Changsha, China
{chromao, wangyi14}@nudt.edu.cn

[3] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
xinyi@ust.hk

Abstract. Order-Revealing Encryption (ORE) provides a practical solution for conducting range queries over encrypted data. Achieving a desirable privacy-efficiency tradeoff in designing ORE schemes has posed a significant challenge. At Asiacrypt 2018, Cash et al. proposed Parameter-hiding ORE (pORE), which specifically targets scenarios where the data distribution shape is known, but the underlying parameters (such as mean and variance) need to be protected. However, existing pORE constructions rely on impractical bilinear maps, limiting their real-world applicability. In this work, we propose an alternative and efficient method for constructing pORE using identification schemes. By leveraging the map-invariance property of identification schemes, we eliminate the need for pairing computations during ciphertext comparison. Specifically, we instantiate our framework with the pairing-free Schnorr identification scheme and demonstrate that our proposed pORE scheme reduces ciphertext size by approximately 31.25% and improves encryption and comparison efficiency by over two times compared to the current state-of-the-art pORE construction. Our work provides a more efficient alternative to existing pORE constructions and could be viewed as a step towards making pORE a viable choice for practical applications.

Keywords: Order-revealing encryption · Property-preserving hash · Identification scheme · Range query

1 Introduction

In recent years, there has been a growing interest in developing cryptographic tools that offer both security and efficiency in supporting computations on encrypted data. This is particularly relevant in light of the increasing need for encrypted databases. Order-Revealing Encryption (ORE) [6, 8] is a special encryption scheme [4] where the ciphertexts reveal the order of the underlying plaintexts. This feature is highly valuable in practice as it allows clients to delegate range queries on encrypted data to (potentially) untrusted servers while ensuring data privacy and confidentiality. In fact, ORE has been utilized in proposals of specific outsourced database systems, including CryptDB [35], Cipherbase [2], and TrustedDB [3]. The practical applications of ORE extend beyond databases and into various other areas, including secure computation, privacy-preserving machine learning, and more. As such, ORE has become an increasingly important tool and has spurred further research.

In an ORE scheme, the order of plaintext is revealed via a comparison algorithm, denoted by Cmp. Specifically, for two ciphertexts c_i and c_j (w.r.t. the messages m_i and m_j), the comparison result $\mathsf{Cmp}(c_i, c_j)$ equals to the predicate function $\mathcal{O}(m_i, m_j)$ defined as:

$$\mathcal{O}(m_i, m_j) = \begin{cases} 1, & \text{if } m_i > m_j \\ -1, & \text{if } m_i < m_j \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

On the Leakage of ORE. Ideally, the "best-possible" security for ORE should reveal nothing about the plaintext but only their order. Such *ideal leakage* profile, denoted by $\mathcal{L}_0$, is defined as follows,

$$\mathcal{L}_0(m_1, \cdots, m_q) := (\forall 1 \leq i, j \leq q, \mathbf{1}(m_i < m_j)). \tag{2}$$

To show such an ideal ORE is achievable in principle, Boneh et al. [8] proposed a construction based on multilinear maps. Unfortunately, current multilinear maps are quite inefficient, and thus such ORE is too impractical for real-world application. Hence, since then there have been various efforts to relax the security requirements of ORE to develop efficient schemes. Below we will introduce several notable constructions that are closely related to our work.

Chenette et al. [12] proposed a much more efficient ORE (CLWW ORE) but with more information leakage than ideal ones. Precisely, in their construction, the most significant differing bit $\mathsf{msdb}(m_i, m_j)$ of plaintexts would be additionally revealed. Their leakage profile, called *CLWW leakage*, is defined as

$$\mathcal{L}_2(m_1, \cdots, m_q) := (\forall 1 \leq i, j \leq q, \mathbf{1}(m_i < m_j), \mathsf{msdb}(m_i, m_j)). \tag{3}$$

More details about the CLWW ORE will be provided in Section 1.2. Lewi and Wu [30] presented an improved version of the above CLWW ORE which leaks strictly less. They constructed a practical, small-domain ORE scheme with best-possible security and extended it to large-domain ORE by block-wise encryption, which leaks the position of the first different block, and thus allows a practitioner to set performance-security tradeoff by tuning the block size.

Unfortunately, a series of work [9, 14, 19, 32] have demonstrated that even hypothetical ideal ORE schemes are insecure for various use cases. This holds

[4] It was called efficiently-orderable encryption in [6].

true even when the scheme only reveals the order of the plaintexts and nothing more. The inherent issue lies in the fact that the mere knowledge of the order of plaintexts can disclose a significant amount of information about the underlying data. For instance, in cases where the data is uniformly selected from the entire domain, even an ideal ORE scheme will inadvertently leak the most significant bits. In fact, the aforementioned attacks reveal that when the adversary possesses a strong estimate of the prior distribution from which the data is drawn, achieving meaningful security is impossible.

Parameter-Hiding ORE. Given the aforementioned attacks against ORE, Cash et al. [10] turned to consider specific scenarios (more details are referred to Appendix A) where the adversary lacks a strong estimate of the prior distribution of the data. For such a case, Cash et al. proposed a new notion, called *parameter-hiding ORE* (pORE), aiming at protecting some information about the underlying data distribution. In particular, Cash et al. [10] mainly consider a ORE scheme with so-called *smoothed CLWW leakage* which is defined as follows:

$$\begin{aligned}\mathcal{L}_1(m_1, \cdots, m_q) :=& (\forall 1 \le i, j, k \le q, \mathbf{1}(m_i < m_j), \\ & \mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_i, m_k))),\end{aligned} \tag{4}$$

and show how to convert it into parameter-hiding ORE by composing with a linear function: for any plaintext m, it is computed as $\alpha m + \beta$ before being encrypted, where α, β are the same across all plaintexts and are sampled as part of the secret key.

Cash et al. [10] proposed a pORE construction using a new primitive called Property-Preserving Hash (PPH). PPH enables the comparison key holder to detect whether the preimage of two hash values satisfies a given predicate $\mathcal{P}$, such as $\mathcal{P}(x, y) = 1$ if $x = y + 1$. A concrete pairing-based instance of PPH was proposed by Cash et al. [10] to construct the pORE scheme. In particular, Cash et al.'s ORE scheme requires $O(4n^2)$ pairings for each comparison between n bits message. To reduce the computational overhead, Cash et al. fixed the random permutation to achieve $O(n)$ comparison overhead, but with a slightly more leakage $\mathcal{L}_1'$ (still smoothed CLWW leakage):

$$\begin{aligned}\mathcal{L}_1'(m_1, \cdots, m_q) :=& (\forall 1 \le i, j, k, l \le q, \mathbf{1}(m_i < m_j), \\ & \mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_k, m_l))).\end{aligned} \tag{5}$$

Compared to $\mathcal{L}_1$, $\mathcal{L}_1'$ reveals extra information, i.e. whether $\mathsf{msdb}(m_i, m_j)$ equals to $\mathsf{msdb}(m_k, m_l))$ even when $i \neq k$. For example, two sets $\{0, 4, 5, 10, 11\}$ and $\{1, 2, 3, 5, 6\}$ are distinguishable with $\mathcal{L}_1'$ leakage since $\mathsf{msdb}(4, 5) = \mathsf{msdb}(10, 11)$ and $\mathsf{msdb}(2, 3) \neq \mathsf{msdb}(5, 6)$. But, two sets $\{0, 4, 5, 10, 11\}$ and $\{1, 2, 3, 4, 5\}$ are also indistinguishable with $\mathcal{L}_1$ leakage.

Motivating Question. As highlighted by Bogatov et al. [4], Cash et al.'s ORE scheme suffers from two limitations that render it impractical. One of these limitations is the use of bilinear maps, which incurs significant computational costs—orders of magnitude higher than other efficient ORE schemes. Motivated by this observation and the specific leakage properties of pORE, our objective is to explore the feasibility of enhancing the efficiency of Cash et al.'s construction while preserving the same level of leakage. Specifically, we aim to address the following question:

Is it possible to design a parameter-hiding order-revealing encryption scheme without pairings?

1.1 Our Contributions

In this work, we provide an affirmative answer to the above question by designing a new pairing-free PPH scheme to improve the efficiency of the state-of-the-art pORE construction [10].

Our main contributions can be summarized as follows.

- We formally define a special type of identification schemes which is of additional property called *map-invariance*, due to which we manage to design a generic construction of PPH schemes. We then formally prove that our PPH construction is restricted-chosen-input secure with respect to the predicate $\mathcal{P}$ where $\mathcal{P}(x, y) = \pm 1$ if and only if $x = y \pm 1$.
- We then provide a generic construction of ORE with smoothed CLWW leakage $\mathcal{L}_1$ from identification protocols, and prove that the proposed scheme is $\mathcal{L}_1$-non-adaptive-simulation secure with respect to the predicate $\mathcal{O}(x, y)$, which is defined as $\mathcal{O}(x, y) = \pm 1$ if and only if $x > y$ or $x < y$. Instanced with Schnorr identification, we presented an ORE scheme, which can be converted to parameter-hiding ORE via Cash et al.'s framework [10].
- We implement our instanced ORE schemes and perform a comprehensive comparison with existing construction (see Table 1). The results demonstrate that our scheme outperforms the current parameter-hiding ORE [10] at the same security level, in which the ciphertext length reduced more than 31.25%, the encryption efficiency increased by nearly 2.6 times and the comparison efficiency increased by more than 3 times.

As also mentioned by Lewi and Wu [30], the leakage inherent in any ORE scheme renders the primitive not always suitable for applications requiring a high level of security. Nevertheless, since it is up to a practitioner to trade off security and performance constraints, we hope our more efficient realization of the pORE scheme could help practitioners make well-informed decisions regarding its suitability for specific applications, especially when the data distribution shape is public, but the underlying parameters (such as mean and variance) need to be protected.

1.2 Technique Overview

First, we provide an overview of our design idea for PPH. To better illustrate how our idea works, we start with recalling Chenette et al.'s construction [12], followed by Cash et al.'s scheme to explain how PPH could be used to reduce the leakage. Thereafter, we illustrate our design idea of pairing-free PPH schemes.

ORE Scheme	Ciphertext size	Encryption cost	Comparison Cost	Leakage
Cash et al.'s ORE [10]	$2n\lvert\mathbb{G}_1\rvert + 2n\lvert\mathbb{G}_2\rvert$	$2n\mathsf{PM}_1 + 2n\mathsf{PM}_2 + 3n\mathsf{PRF}$	$4n^2\mathsf{BP}$	$\mathcal{L}_1$
Cash et al.'s ORE* [10]	$2n\lvert\mathbb{G}_1\rvert + 2n\lvert\mathbb{G}_2\rvert$	$2n\mathsf{PM}_1 + 2n\mathsf{PM}_2 + 3n\mathsf{PRF}$	$4n\mathsf{BP}$	$\mathcal{L}'_1$
Our Sch-ORE	$4n\lvert\mathbb{G}_1\rvert + 3n\lvert\mathbb{Z}_p\rvert$	$6n\mathsf{PM}_1 + 5n\mathsf{PRF}$	$8n^2\mathsf{PM}_1 + 2n\mathsf{PRF}$	$\mathcal{L}_1$
Our Sch-ORE*	$4n\lvert\mathbb{G}_1\rvert + 3n\lvert\mathbb{Z}_p\rvert$	$6n\mathsf{PM}_1 + 5n\mathsf{PRF}$	$8n\mathsf{PM}_1 + 2n\mathsf{PRF}$	$\mathcal{L}'_1$

Table 1. Comparison with existing ORE schemes with smoothed CLWW leakage. λ is the security parameter, n is the bit-length of the plaintext; $|\mathbb{G}_1|$ and $|\mathbb{G}_2|$ is the size of elements in groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively; PM_1, PM_2 and BP is the group exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$ and the bilinear pairing operation, respectively; PRF is the pseudorandom function operation. ORE* refers to the ORE scheme with fixed permutation.

CLWW ORE [12]. In CLWW ORE scheme [12], the encryption process involves splitting a n-bits plaintext m into n values by concatenating each bit of m with all more significant bits, and the resulting value is then processed through a keyed pseudorandom function F with the secret key k. The output of the PRF is numerically added to the next less significant bit. More precisely, the resulting ciphertext c consists of $\{u_1, \cdots, u_n\}$:

$$u_i = \mathcal{E}(k, m, i) = F(k, i||m_{[:i-1]}||0^{n-i+1}) + m_{[i]} \mod M, \forall\ i \in [n]$$

where $m_{[i]}$ denotes the i-th bit of m, $m_{[:i]}$ denotes the first i-bits of m and 0^i denotes the i-length zero bit-string [5]. To compare two ciphertexts, the comparator needs to find the smallest index i at which the two sequences occur different values. Denote the ciphertext w.r.t. m' as $c' = \{u'_1, \cdots, u'_n\}$. Then, the comparator judges $m > m'$ if $u_i = u'_i + 1 \mod M$ (or $m < m'$ if $u_i = u'_i - 1 \mod M$). If not found, it means $m = m'$. Obviously, the first index i for which $u_i = u'_i \pm 1$ leaks the information $\mathsf{msdb}(m, m')$.

Reducing CLWW ORE Leakage via PPH. The scheme by Cash et al. [10] improves the leakage of the above CLWW ORE. Their scheme builds upon the CLWW construction, but incorporates a permutation step for the list of PRF outputs, as finding a pair that differs by one is sufficient. However, solely permuting the outputs is not adequate for reducing leakage, as an adversary can still deduce the number of common elements between two ciphertexts.

To the end, Cash et al. [10] introduced a new primitive, called *property-preserving hash* (PPH) to "hash" the elements of the CLWW ORE ciphertexts before outputting them. The most essential property of PPH is the randomized output which means that the same element hashed twice will result in different hashes, due to which the adversary is unable to determine the number of common elements between two ciphertexts, thereby preventing the identification of the location of the differing bit.

[5] In Chenette et al.'s work, they took M to be the minimum value, i.e. $M = 3$, making the ciphertext size only $\lceil n \cdot \log_2 3 \rceil$ bits.

Pairing-Based PPH [10]. The main difficulty of constructing PPH lies in the fact that how to randomize the hash output while still preserve the main property of the input. In [10], Cash et al. proposed a concrete construction of PPH based on bilinear pairings. Precisely, for each input u_i, the hash output consists of the following elements

$$\left(g_1^{r_0}, g_1^{r_0 \cdot H(s,u_i)}, g_2^{r_1}, g_2^{r_1 \cdot H(s,u_i+1)}\right)$$

where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear pairing, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are the generators of the prime order q. One could note that due to two freshly chosen randomness $r_0, r_1 \in \mathbb{Z}_q^*$, both u_i and $u_i + 1$ are hidden in the uniformly distributed hash outputs. As shown in Figure 1, for another input u'_j which has the hash output:

$$\left(g_1^{r'_0}, g_1^{r'_0 \cdot H(s,u'_j)}, g_2^{r'_1}, g_2^{r'_1 \cdot H(s,u'_j+1)}\right),$$

the comparison algorithm computes

$$e\left(g_1^{r_0 \cdot H(s,u_i)}, g_2^{r'_1}\right) \overset{?}{=} e\left(g_1^{r_0}, g_2^{r'_1 \cdot H(s,u'_j+1)}\right)$$

to test whether $u_i = u'_j + 1$. A similar computation could be performed to test whether $u_i = u'_j - 1$ or not.

Our Efficient PPH from Schnorr Identification. In this work, we provide a new approach to construct PPH. In the following description, we consider the predicate $\mathcal{P}$ defined as follows:

$$\mathcal{P}(x, y) = \begin{cases} 1, & \text{if } x = y + 1 \\ -1, & \text{if } x = y - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

For a more intuitive understanding, here we illustrate our main idea by introducing the concrete PPH based on Schnorr identification. Figure 1 depicts our core idea via a rough comparison with Cash et al.'s pairing-based PPH construction.

Schnorr Identification. The Schnorr scheme is simply described as: Given an cyclic group $\mathbb{G} = \langle g \rangle$ of the prime order p, the secret key is $\mathsf{sk} := x \leftarrow_{\$} \mathbb{Z}_p^*$ and the public key is $\mathsf{pk} := (g, y = g^x) \in \mathbb{G}^2$. The prover generates the commitment $\mathsf{cmt} := w = g^{r_0}$ with a random integer $r_0 \in \mathbb{Z}_p^*$ and the response $\mathsf{rsp} := z = r_0 - \xi \cdot x$ where $\xi \in \mathbb{Z}_p^*$ is the verifier's challenge. The verifier checks whether $w = g^z \cdot y^\xi$ holds to decide acceptance or rejection.

A New PPH without Pairings. In our PPH, the encryptor (who computes PPH in the underlying ORE) generates two key-pairs $(x_0, y_0 := g^{x_0})$ and $(x_1, y_1 := g^{x_1})$, hides u_i into the identification response $z_0 := H(s, u_i) \cdot r_0 - x_0 \cdot \xi$ and $z_1 := H(s, u_i+1) \cdot r_1 - x_1 \cdot \xi$ with the same challenge $\xi \leftarrow_{\$} \mathbb{Z}_p^*$ and two randomness $r_0, r_1 \leftarrow_{\$} \mathbb{Z}_p^*$ respectively. Then, the encryptor computes the output of the PPH:

$$h_i := \{g^{r_1}, y_0^{r_1}, z_0, \xi, g^{r_0}, y_1^{r_0}, z_1\}, \text{w.r.t. } u_i. \tag{7}$$

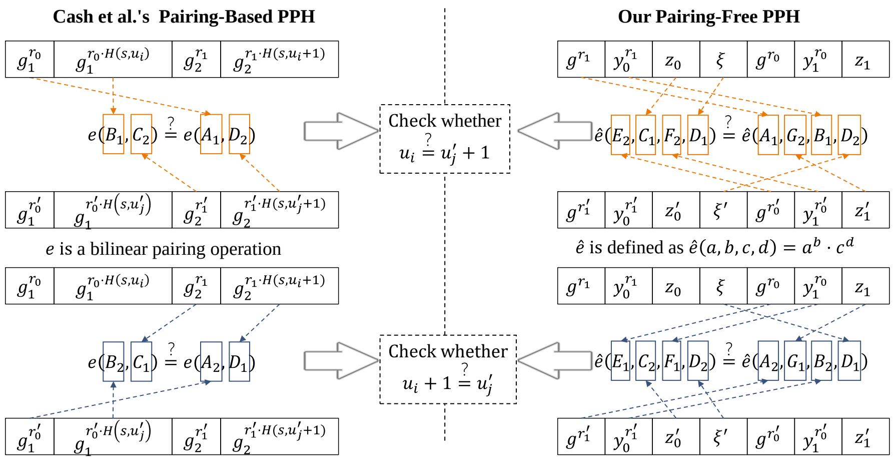

Cash et al.'s Pairing-Based PPH
Our Pairing-Free PPH
Check whether $u_i \overset{?}{=} u'_j + 1$
Check whether $u_i + 1 \overset{?}{=} u'_j$
$e(B_1, C_2) \overset{?}{=} e(A_1, D_2)$
$e(B_2, C_1) \overset{?}{=} e(A_2, D_1)$
$\hat{e}(E_2, C_1, F_2, D_1) \overset{?}{=} \hat{e}(A_1, G_2, B_1, D_2)$
$\hat{e}(E_1, C_2, F_1, D_2) \overset{?}{=} \hat{e}(A_2, G_1, B_2, D_1)$
e is a bilinear pairing operation
$\hat{e}$ is defined as $\hat{e}(a,b,c,d) = a^b \cdot c^d$
$g_1^{r_0}$ $g_1^{r_0 \cdot H(s,u_i)}$ $g_2^{r_1}$ $g_2^{r_1 \cdot H(s,u_i+1)}$
$g_1^{r'_0}$ $g_1^{r'_0 \cdot H(s,u'_j)}$ $g_2^{r'_1}$ $g_2^{r'_1 \cdot H(s,u'_j+1)}$
g^{r_1} $y_0^{r_1}$ z_0 ξ g^{r_0} $y_1^{r_0}$ z_1
$g^{r'_1}$ $y_0^{r'_1}$ z'_0 ξ' $g^{r'_0}$ $y_1^{r'_0}$ z'_1

Fig. 1. Visual description of two PPHs.

For comparison with $h'_j := \{g^{r'_1}, y_0^{r'_1}, z'_0, \xi', g^{r'_0}, y_1^{r'_0}, z'_1\}$ w.r.t. u'_j, one can utilize the commitment recovery algorithm to check the predicate $\mathcal{P}(u_i, u'_j)$ as follows:

$$(g^{r'_1})^{z_0} \cdot (y_0^{r'_1})^{\xi} \overset{?}{=} (g^{r_0})^{z'_1} \cdot (y_1^{r_0})^{\xi'},$$

to test whether $u_i = u'_j + 1$, equivalent to check $g^{r_0 \cdot r'_1 \cdot H(s,u_i)} \overset{?}{=} g^{r_0 \cdot r'_1 \cdot H(s,u'_j+1)}$. A similar computation could be performed to test whether $u_i = u'_j - 1$ or not.

In fact, the value u_i is hidden in two responses z_0 and z_1 with independent random integers r_0 and r_1. Based on the security of Schnorr identification, it is known that r_0, r_1, x_0, x_1 would not be revealed. Thus, the security of PPH holds. Based on the binding property of commitments, one could determine that two equal commitment values are related to the same message.

Generalization from Specific Identifications. Inspired by the above concrete PPH construction, we formalize a new property named *map-invariance* for identification protocols, and show how to generically construct PPH from specific identification with map-invariance. Thereafter, we show that our generic PPH construction achieves restricted-chosen-input security which is crucial for ORE schemes with smoothed CLWW leakage that could be extended to parameter-hiding ORE schemes with Cash et al.'s approach [10, Section 4].

1.3 Related Work

In this section, we survey some literature on order-revealing and order-preserving encryption, as well as the existing work on secure range query protocols.

Order-Preserving Encryption. Prior to ORE, Agrawal et al. [1] first introduced the notion of *order-preserving encryption (OPE)*, which can encrypt numeric data and order ciphertexts with the numerical comparison operator.

Subsequently, Boldyreva et al. [5] formalized the ideal security definition for OPE, which states that ciphertexts should only reveal the order of plaintexts without any other information leakages, similar to the ideal security definition for ORE. However, Boldyreva et al. [5] showed that this "best-possible" security cannot be achieved in stateless and immutable OPE schemes. They also showed that any OPE scheme with ideal security will make the ciphertext length grow exponentially with the plaintext length. Since then, a number of OPE schemes have been proposed such as [6,23,24,34,36]. To achieve the ideal security, Popa et al. [34] proposed a mutable OPE scheme that constructs a stateful binary tree on plaintexts and an interactive protocol for path-based ciphertext updates. Unfortunately, due to the constraint on the ciphertext space, most OPE constructions are vulnerable to various inference attacks, as demonstrated by Naveed et al. [32], where attackers can recover partial bits of plaintexts.

Secure Range Query Protocols. From OPE/ORE schemes, the general idea is to modify all comparison operators to the ORE comparison algorithm in some data structures optimized for range queries, e.g., the B+ tree working on top of an ORE scheme [4]. In ORE/OPE-based applications, one mostly considers the snapshot attacker [19, 32], who can observe all the database contents at one time instant. This makes designers more inclined to focus on leakage from fixed datasets and minimize the information available to adversaries through reconstruction. Undisputedly, leakage from adaptive queries can lead to additional privacy issues for clients and servers. In this scenario, the adversarial model mainly discusses leakage-abuse attacks based on different datasets, such as those based on access-pattern leakage [18, 22, 29], search-pattern leakage [27, 31], volumetric leakage [17, 22]. Approaches that help protect against such an attacker include searchable symmetric encryption (SSE) [13, 15], Oblivious RAM [11], response-hiding constructions [21, 28]. These schemes are more secure than OPE/ORE schemes, although they still leak some information, and in general, are more expensive to compute. As shown in Bogatov et al.'s comparative evaluation [4] of ORE and other secure range-query protocols, the ORE-based approach is provably I/O optimal and can potentially be extended by using another data structure with ORE.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ be the security parameter and write $\mathsf{negl}(\lambda)$ to denote a negligible function w.r.t. λ and $\mathsf{poly}(\lambda)$ to denote a polynomial function w.r.t. λ. Let $[n]$ denote the set of integers $\{1, 2, \cdots, n\}$ for $n \in \mathbb{N}$. For a bit string $b = b_1 b_2 \cdots b_n$, let $b_{[:i]} = b_1 b_2 \cdots b_i$ denote the first i bits of b in the bit string representing and 0^i denote a i-length zero string in the bit string representing. For two bit strings b and b', we write $\mathsf{msdb}(b, b')$ to denote the most significant different bit of them. For a given set U, we let $u \leftarrow_{\$} U$ denote the uniform sampling from U. For any bit strings $x, y \in \{0,1\}^*$, we write $x||y$ to denote the concatenation of x and y. If $\mathcal{P}$ is a predicate on x, $\mathbf{1}(\mathcal{P}(x))$ means the indicator function for $\mathcal{P}$, that is $\mathbf{1}(\mathcal{P}(x)) = 1$ if and only if $\mathcal{P}(x) = 1$.

2.1 Keyed Hash Function

Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a keyed hash function family [20]. Here $\mathcal{K}$ is the key space of F, $\mathcal{X}$ is the domain of F and $\mathcal{Y}$ is the range of F.

Definition 1 (Collision Resistance). *Given a fixed security parameter λ, a keyed hash function family $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is collision resistant if for any efficient adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{\mathsf{COL}}_{F,\mathcal{A}}(\lambda) := \Pr\left[(x, x') \leftarrow \mathcal{A} : F(k, x) = F(k, x') \wedge x \neq x'\right]$$

is negligible.

Definition 2 (Pseudorandom [16]). *Given a fixed security parameter λ, a keyed hash function family $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is pseudorandom if for any efficient adversary $\mathcal{A}$, its advantage*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{PRF}}_{F,\mathcal{A}}(\lambda) := \big|&\Pr\left[k \leftarrow_\$ \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(\lambda) = 1\right] - \\ &\Pr\left[f \leftarrow_\$ \mathsf{Funs}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)}(\lambda) = 1\right]\big|\end{aligned}$$

is negligible, where $\mathsf{Funs}(\mathcal{X}, \mathcal{Y})$ is the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

Definition 3 (Entropy Smoothing). *Given a fixed security parameter λ, a keyed hash function family $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is entropy smoothing if for any efficient adversary $\mathcal{A}$, its advantage*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ES}}_{F,\mathcal{A}}(\lambda) := |&\Pr\left[k \leftarrow_\$ \mathcal{K}, \delta \leftarrow_\$ \mathcal{X} : \mathcal{A}(\lambda, k, F(k, \delta)) = 1\right] - \\ &\Pr\left[k \leftarrow_\$ \mathcal{K}, h \leftarrow_\$ \mathcal{Y} : \mathcal{A}(\lambda, k, h) = 1\right]|\end{aligned}$$

is negligible.

2.2 Property-Preserving Hash

In this section, we recall the syntax and security of a basic tool, called *Property-Preserving Hash (PPH)* [10], which is essentially the simplified variant of property-preserving encryption schemes [33]. In the PPH scheme, a hash key hk is a necessary input to calculate the hash value. The difference is that a test key tk can be used to determine whether two different hash values $\{h, h'\}$ with respect to two distinct messages $\{x, y\}$ satisfy an associated property $\mathcal{P}$.

Definition 4 (Property-Preserving Hash). *A property-preserving hash scheme is a tuple of polynomial-time algorithms $\Gamma = (\mathsf{PPH.KeyGen}, \mathsf{PPH.Hash}, \mathsf{PPH.Test})$ defined as follows:*

- $(\mathsf{par}, \mathsf{hk}, \mathsf{tk}) \leftarrow \mathsf{PPH.KeyGen}(1^\lambda)$*: The key generation algorithm is a randomized algorithm that takes as input a security parameter λ and outputs the system parameters* par*, the hash key* hk *and the test key* tk*. These implicitly define the input domain* DSet *and the output range* HSet *of the hash algorithm.*

- $h \leftarrow \mathsf{PPH.Hash}(\mathsf{hk}, u)$*: The hash evaluation algorithm is a randomized algorithm that takes as input a hash key* hk *and a message* $u \in \mathsf{DSet}$*, and outputs a hash value* $h \in \mathsf{HSet}$*.*
- $b \leftarrow \mathsf{PPH.Test}(\mathsf{tk}, h, h')$*: The hash test algorithm is a deterministic algorithm that takes as input a test key* tk *and two hash values* $(h, h') \in \mathsf{HSet}^2$ *and outputs a bit* $b \in \{0, 1\}$*.*

Correctness of PPH Schemes. For a predicate $\mathcal{P}$, the PPH scheme Γ is *computationally correct* if the following advantage

$$\mathsf{Adv}^{\mathsf{COR}}_{\Gamma,\mathcal{P},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{c|l} \mathsf{Test}(\mathsf{tk}, h, h') & (\mathsf{par},\mathsf{hk},\mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \neq \mathcal{P}(x, y) & x, y \leftarrow \mathcal{A}^{\mathsf{Hash}(\mathsf{hk},\cdot)}(\mathsf{tk}), \\ & h \leftarrow \mathsf{Hash}(\mathsf{hk}, x), \\ & h' \leftarrow \mathsf{Hash}(\mathsf{hk}, y), \end{array}\right]$$

is negligible for any efficient PPT adversary $\mathcal{A}$ which can query the Hash oracle with any inputs in DSet.

Security of PPH Schemes. For a predicate $\mathcal{P}$, the PPH scheme Γ is *restricted-chosen-input secure* if the following advantage

$$\mathsf{Adv}^{\mathsf{PPH}}_{\Gamma,\mathcal{P},\mathcal{A}}(\lambda) := 2\Pr\left[\begin{array}{c|l} & (\mathsf{par},\mathsf{hk},\mathsf{tk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ & x^* \leftarrow \mathcal{A}(\mathsf{tk}), \\ b = b' & h_0 \leftarrow \mathsf{Hash}(\mathsf{hk}, x^*), \\ & h_1 \leftarrow_\$ \mathsf{HSet}, b \leftarrow_\$ \{0, 1\}, \\ & b' \leftarrow \mathcal{A}^{\mathsf{Hash}(\mathsf{hk},\cdot)}(\mathsf{tk}, x^*, h_b), \end{array}\right] - 1$$

is negligible for any efficient PPT adversary $\mathcal{A}$ which can query the Hash oracle with restricted inputs $x \in \mathsf{DSet}$ satisfying $\mathcal{P}(x, x^*) = 0$.

2.3 Parameter-Hiding ORE

In this part, we review the syntax and security of parameter-hiding ORE (pORE) schemes formalized by Cash et al. [10].

Definition 5 (Parameter-Hiding ORE). *For a predicate* $\mathcal{O}$ *defined over a well-ordered domain* $\mathcal{D}$*, a parameter-hiding ORE scheme is a tuple of polynomial-time algorithms* $\Pi = (\mathsf{ORE.KGen}, \mathsf{ORE.Enc}, \mathsf{ORE.Cmp})$ *defined as follows:*

- $(\mathsf{par}, \mathsf{msk}, \mathsf{ck}) \leftarrow \mathsf{ORE.KGen}(1^\lambda)$*: The key generation algorithm is a randomized algorithm that takes as input a security parameter* λ *and outputs the system parameters* par*, the master secret key* msk *and the comparison key* ck*. We remark* par *is an implicit input of following algorithms.*
- $c \leftarrow \mathsf{ORE.Enc}(\mathsf{msk}, m)$*: The encryption algorithm is a randomized algorithm that takes as input the master secret key* msk *and a message* $m \in \mathcal{D}$*, and outputs a ciphertext* c*.*
- $b \leftarrow \mathsf{ORE.Cmp}(\mathsf{ck}, c, c')$*: The comparison algorithm is a deterministic algorithm that takes as input the comparison key* ck *and two ciphertexts* c *(w.r.t. the plaintext* m*) and* c' *(w.r.t. the plaintext* m'*), and outputs a flag* b*.*

Correctness of ORE schemes. For a predicate $\mathcal{O}$, the ORE scheme Π is *computationally correct* if the following advantage

$$\mathsf{Adv}^{\mathsf{COR}}_{\Pi,\mathcal{O},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{r} \mathsf{Cmp}(\mathsf{ck}, c, c') \\ \neq \mathcal{O}(m, m') \end{array}\middle| \begin{array}{l} (\mathsf{par}, \mathsf{msk}, \mathsf{ck}) \leftarrow \mathsf{KGen}(1^\lambda) \\ c \leftarrow \mathsf{Enc}(\mathsf{msk}, m) \\ c' \leftarrow \mathsf{Enc}(\mathsf{msk}, m') \end{array}\right]$$

is negligible for any efficient adversary $\mathcal{A}$.

Security of ORE schemes. The ORE scheme is non-adaptive simulation-based secure with the leakage function $\mathcal{L}(\cdot)$, if there exists a polynomial-size simulator $\mathcal{S}$ to construct a non-adaptive experiment $\mathsf{Sim}^{\mathsf{ORE}}_{\mathcal{A},\mathcal{L},\mathcal{S}}(\lambda)$ which is computationally indistinguishable from the real non-adaptive experiment $\mathsf{Real}^{\mathsf{ORE}}_{\mathcal{A}}(\lambda)$ for any PPT adversary $\mathcal{A}$. Experiments are described in Fig. 2.

```
$\mathsf{Real}^{\mathsf{ORE}}_{\mathcal{A}}(\lambda)$:
1: $(\mathsf{par}, \mathsf{msk}, \mathsf{ck}) \leftarrow \mathsf{ORE.KGen}(1^\lambda)$;
2: $(m_1, \cdots, m_q) \leftarrow \mathcal{A}$;
3: for $1 \le i \le q$ do
4:     $c_i \leftarrow \mathsf{ORE.Enc}(\mathsf{msk}, m_i)$;
5: return $(c_1, \cdots, c_q)$;

$\mathsf{Sim}^{\mathsf{ORE}}_{\mathcal{A},\mathcal{L},\mathcal{S}}(\lambda)$:
1: $\mathsf{st}_{\mathcal{S}} \leftarrow \mathcal{S}(\lambda)$;
2: $(m_1, \cdots, m_q) \leftarrow \mathcal{A}$;
3: $(c_1, \cdots, c_q) \leftarrow \mathcal{S}(\mathsf{st}_{\mathcal{S}}, \mathcal{L}(m_1, \cdots, m_q))$;
4: return $(c_1, \cdots, c_q)$;
```

Fig. 2. Experiments $\mathsf{Real}^{\mathsf{ORE}}_{\mathcal{A}}(\lambda)$ and $\mathsf{Sim}^{\mathsf{ORE}}_{\mathcal{A},\mathcal{L},\mathcal{S}}(\lambda)$.

3 Identification Schemes With Map-Invariance

In this section, we recall the definition of canonical identification schemes described in [26]. Then, we formalize a type of identification protocols with map-invariance property and give an instance from Schnorr Identification.

3.1 Formal Definitions

Definition 6 (Canonical Identification Scheme). *A canonical identification scheme* ID *is defined as a tuple of algorithms* $\mathsf{ID} := (\mathsf{Setup}, \mathsf{IGen}, \mathsf{P}, \mathsf{V})$ *with system parameter space* $\mathcal{SP}$*, public key space* $\mathcal{PK}$*, commitment space* $\mathcal{W}$*, randomness space* $\mathcal{R}$*, challenge space* $\mathcal{CH}$ *and response space* $\mathcal{Z}$*.*

- $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$*: Taking security parameter* λ *as input, it outputs system parameters* $\mathsf{par} \in \mathcal{SP}$*.*
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$*: Taking system parameters* par *as input, it outputs a secret and public key pair* $(\mathsf{sk}, \mathsf{pk})$*.*
- *The prover algorithm* $\mathsf{P} = (\mathsf{Com}, \mathsf{Rsp})$ *consists of two algorithms:*
  - $(\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}, \mathsf{sk}; \mathsf{rnd})$*: Taking system parameters* par *and secret key* sk *as inputs, it outputs commitment* $\mathsf{cmt} \in \mathcal{W}$ *and state* st*. Here we explicitly write the randomness* $\mathsf{rnd} \in \mathcal{R}$ *in the input.*
  - $\mathsf{rsp} \leftarrow \mathsf{Rsp}(\mathsf{par}, \mathsf{sk}, \mathsf{st}, \mathsf{ch})$*: Taking system parameters* par*, secret key* sk*, state* st *and challenge* $\mathsf{ch} \in \mathcal{CH}$ *as inputs, it outputs a response* rsp*.*
- $b \leftarrow \mathsf{V}(\mathsf{par}, \mathsf{pk}, \mathsf{tr})$*: Taking system parameters* par*, the public key* pk *and the transcript* $\mathsf{tr} = (\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$ *as inputs, and outputs a bit* $b \in \{0, 1\}$*. If the verifier accepts,* $b = 1$*; Otherwise,* $b = 0$*.*

The identification works as follows: The prover first publishes its public key pk and sends a commitment cmt to the verifier. Then, the verifier randomly chooses a challenge ch from the challenge space $\mathcal{CH}$ and sends it to the prover. After receiving the challenge, the prover computes a response rsp and sends it to the verifier. Finally, the verifier makes a decision (acceptance or rejection) based on the public key and the conversation transcript.

Below, we discuss some properties of commitments in special identification schemes, which are crucial to our efficient PPH constructions.

Definition 7 (Commitment-Independency). *An identification scheme* ID *is commitment-independent if for* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$*,* $(\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}, \mathsf{sk}; \mathsf{rnd})$ *and any* $\mathsf{rnd}, \mathsf{rnd}' \in \mathcal{R}$*, following conditions holds:*

- *the value of* $(\mathsf{cmt}, \mathsf{st})$ *is uniquely determined by system parameter* par *and randomness* rnd*, and independent of the secret key* sk *and the public key* pk*;*
- $\mathsf{Com}_1(\mathsf{par}, \mathsf{sk}; \mathsf{rnd}) = \mathsf{Com}_1(\mathsf{par}, \mathsf{sk}; \mathsf{rnd}')$ *if and only if* $\mathsf{rnd} = \mathsf{rnd}'$*, where algorithm* Com_1 *only returns the first element of the output of algorithm* Com *(i.e.,* cmt*).*

Definition 8 (Commitment-Recoverability [25]). *An identification scheme* ID *is commitment-recoverable if for* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$*, any* $\mathsf{ch} \in \mathcal{CH}$*, and any* $\mathsf{rsp} \in \mathcal{Z}$*, there exists a unique commitment* $\mathsf{cmt} \in \mathcal{W}$ *such that* $\mathsf{V}(\mathsf{par}, \mathsf{pk}, \mathsf{cmt}, \mathsf{ch}, \mathsf{rsp}) = 1$*. And, the unique commitment* cmt *can be publicly computed by a commitment recovery algorithm* Rec*, i.e.,*

$$\mathsf{cmt} = \mathsf{Rec}(\mathsf{par}, \mathsf{pk}, \mathsf{ch}, \mathsf{rsp}).$$

Definition 9 (Commitment-Augmentability [7]). *An identification scheme* ID *is augmentable if for* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$ *and* $(\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}, \mathsf{sk}; \mathsf{rnd})$*, there exists an* append algorithm Apd *that takes as input* cmt *and randomness* $\mathsf{rnd}' \in \mathcal{R}$ *and outputs* cmt' *satisfying*

$$\mathsf{cmt}' \leftarrow \mathsf{Com}_1(\mathsf{par}, \mathsf{sk}; \mathsf{rnd} \cdot \mathsf{rnd}'),$$

where $\cdot$ *is the operation defined over* $\mathcal{R}$*.*

Definition 10 (Response-Indistinguishability). *An identification scheme* ID *is response-indistinguishable if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ winning the game* $\mathsf{IND}^{\mathsf{RSP}}_{\mathsf{ID},\mathcal{A}}(\lambda)$*, defined as* $\mathsf{Adv}^{\mathsf{RSP}}_{\mathsf{ID},\mathcal{A}}(\lambda) = 2\Pr\left[\mathsf{IND}^{\mathsf{RSP}}_{\mathsf{ID},\mathcal{A}}(\lambda) = 1\right] - 1$*, is negligible.*

```
Game $\mathsf{IND}^{\mathsf{RSP}}_{\mathsf{ID},\mathcal{A}}(\lambda)$ :
1: $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$; $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$; $\mathsf{rnd} \leftarrow_\$ \mathcal{R}$;
2: $(\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}, \mathsf{sk}; \mathsf{rnd})$; $\mathsf{ch} \leftarrow_\$ \mathcal{CH}$;
3: $\mathsf{rsp}_0 \leftarrow \mathsf{Rsp}(\mathsf{par}, \mathsf{sk}, \mathsf{st}, \mathsf{ch})$; $\mathsf{rsp}_1 \leftarrow_\$ \mathcal{Z}$;
4: $b \leftarrow_\$ \{0, 1\}; b' \leftarrow \mathcal{A}(\mathsf{par}, \mathsf{sk}, \mathsf{ch}, \mathsf{rsp}_b)$
5: return $b \overset{?}{=} b'$
```

Fig. 3. Game $\mathsf{IND}^{\mathsf{RSP}}_{\mathsf{ID},\mathcal{A}}(\lambda)$ for indistinguishability on response.

If the commitment-independency property holds, the commit algorithm can be denoted as $(\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}; \mathsf{rnd})$. If the commitment-recoverability property holds, the verifier algorithm V checks whether cmt equals to $\mathsf{Rec}(\mathsf{par}, \mathsf{pk}, \mathsf{ch}, \mathsf{rsp})$.

We now define a new property, called map-invariance, meaning that the verification equation $\mathsf{cmt} = \mathsf{Rec}(\mathsf{par}, \mathsf{pk}, \mathsf{ch}, \mathsf{rsp})$ holds even after permuting the public key and the commitment using the same randomness.

Definition 11 (Map-Invariance). *An identification scheme* ID *is of map-invariance if following conditions hold:*

- *The identification scheme* ID *is correct, commitment-independent, commitment-recoverable, commitment-augmentable and response-indistinguishable.*
- *There exist two mapping algorithms* ParMap *that takes as input* par *and* rnd' *and outputs* rpar*, and* PkMap *that takes as input* pk *and* rnd' *and outputs* rpk*, such that*

$$\Pr\left[\mathsf{Apd}(\mathsf{cmt}, \mathsf{rnd}') \neq \mathsf{Rec}(\mathsf{rpar}, \mathsf{rpk}, \mathsf{ch}, \mathsf{rsp}) \middle| \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par}), \\ \mathsf{rnd}, \mathsf{rnd}' \leftarrow_\$ \mathcal{R}, \mathsf{ch} \leftarrow_\$ \mathcal{CH}, \\ (\mathsf{cmt}, \mathsf{st}) \leftarrow \mathsf{Com}(\mathsf{par}; \mathsf{rnd}), \\ \mathsf{rsp} \leftarrow \mathsf{Rsp}(\mathsf{par}, \mathsf{sk}, \mathsf{st}, \mathsf{ch}), \\ \mathsf{rpar} \leftarrow \mathsf{ParMap}(\mathsf{par}, \mathsf{rnd}'), \\ \mathsf{rpk} \leftarrow \mathsf{PkMap}(\mathsf{pk}, \mathsf{rnd}') \end{array}\right]$$

is negligible, and for $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{IGen}(\mathsf{par})$ *and* $\mathsf{rnd}' \leftarrow_\$ \mathcal{R}$*, following two distributions are computationally indistinguishable.*

$$\left\{(\mathsf{par}, \mathsf{pk}, \mathsf{rpar}_0, \mathsf{rpk}_0) \middle| \begin{array}{l} \mathsf{rpar}_0 \leftarrow \mathsf{ParMap}(\mathsf{par}, \mathsf{rnd}') \\ \mathsf{rpk}_0 \leftarrow \mathsf{PkMap}(\mathsf{pk}, \mathsf{rnd}') \end{array}\right\},$$

$$\left\{(\mathsf{par}, \mathsf{pk}, \mathsf{rpar}_1, \mathsf{rpk}_1) \middle| \begin{array}{l} \mathsf{rpar}_1 \leftarrow_\$ \mathcal{SP} \\ \mathsf{rpk}_1 \leftarrow_\$ \mathcal{PK} \end{array}\right\}.$$

3.2 An Instance from Schnorr Identification

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group with the generator g of the prime order p. For the IGen execution, it randomly selects an integer $\mathsf{sk} := x \leftarrow_\$ \mathbb{Z}_p^*$ and computes the public key $\mathsf{pk} := y = g^x \in \mathbb{G}$. The prover generates a commitment $\mathsf{cmt} := w = g^r \in \mathbb{G}$ with a random integer $\mathsf{rnd} := r \leftarrow_\$ \mathbb{Z}_p^*$. Then, the verifier picks up a random challenge $\mathsf{ch} := e \leftarrow_\$ \mathbb{Z}_p$. The prover computes the response $\mathsf{rsp} := z = r - e \cdot x \mod p$. The verifier checks whether $w = g^z \cdot y^e$ holds. The algorithms $\mathsf{Com}, \mathsf{Rsp}, \mathsf{Rec}$ are defined as follows:

```
Schnorr Identification:
 1: $\mathsf{par} := g \leftarrow_\$ \mathbb{G}, \mathsf{sk} := x \leftarrow_\$ \mathbb{Z}_p^*, \mathsf{pk} := y = g^x \in \mathbb{G};$
 2: $(\mathsf{cmt}, \mathsf{st}) := (w = g^r, r) \leftarrow \mathsf{Com}(\mathsf{par}; \mathsf{rnd})$
 3: $\mathsf{ch} := e \leftarrow_\$ \mathbb{Z}_p$
 4: $\mathsf{rsp} := z \leftarrow \mathsf{Rsp}(\mathsf{par}, \mathsf{sk}, \mathsf{st}, \mathsf{ch}) = r - e \cdot x \mod p$
 5: $\mathsf{cmt}' := w' \leftarrow \mathsf{Rec}(\mathsf{par}, \mathsf{pk}, \mathsf{ch}, \mathsf{rsp}) = g^z \cdot y^e \in \mathbb{G}$
```

Theorem 1. *The Schnorr identification is of map-invariance under the decisional Diffie-Hellman (DDH) assumption.*

Proof. One can note that the commitment w is independent of the secret key x and $g^r = g^{r'}$ if and only if $r = r'$ for any $r, r' \in \mathbb{Z}_p^*$. The commitment space is the group $\mathbb{G}$ of prime order p and the challenge space is $\mathbb{Z}_p$. So, only one group element can be recovered by Rec with the input instance (g, y, e, z) and equals to the prover's commitment w.

With a new randomness $\mathsf{rnd}' := r'$, the maps Apd, ParMap and PkMap are defined as follows:

$$\begin{aligned} w' := w^{r'} &\leftarrow \mathsf{Apd}(w, r'), \\ \mathsf{rpar} = g^{r'} &\leftarrow \mathsf{ParMap}(g, r'), \\ \mathsf{rpk} = y^{r'} &\leftarrow \mathsf{PkMap}(y, r'). \end{aligned} \tag{8}$$

Obviously, the output of algorithm Apd can be derived from $\mathsf{Com}(\mathsf{par}; \mathsf{rnd} \cdot \mathsf{rnd}')$. We claim that $\mathsf{Apd}(\mathsf{cmt}, \mathsf{rnd}') = \mathsf{Rec}(\mathsf{rpar}, \mathsf{rpk}, \mathsf{ch}, \mathsf{rsp})$ holds with overwhelming probability, as $\mathsf{Apd}(\mathsf{cmt}, \mathsf{rnd}') = g^{r \cdot r'}$ and $\mathsf{Rec}(\mathsf{rpar}, \mathsf{rpk}, \mathsf{ch}, \mathsf{rsp}) = g^{r' \cdot z} y^{r' \cdot e} = g^{r \cdot r'}$.

Since r is uniformly distributed in $\mathbb{Z}_p^*$, the response $\mathsf{rsp}_0 := z_0 = r - e \cdot x$ is also uniformly distributed in $\mathbb{Z}_p^*$. While $\mathsf{rsp}_1 := z_1 \leftarrow_\$ \mathbb{Z}_p^*$ is uniformly sampled from $\mathbb{Z}_p^*$, no adversary can distinguish z_0 and z_1 with non-negligible advantage.

Finally, if a PPT adversary $\mathcal{A}$ can distinguish $(\mathsf{par}, \mathsf{pk}, \mathsf{rpar}_0, \mathsf{rpk}_0)$ and $(\mathsf{par}, \mathsf{pk}, \mathsf{rpar}_1, \mathsf{rpk}_1)$ with the probability ϵ, we can build an adversary $\mathcal{B}$ to break the DDH problem as follows: Let (g, g^a, g^b, Z) be a DDH instance, $\mathcal{B}$ simulates two tuples as

$$\mathsf{par} = g, \mathsf{pk} = g^a, \mathsf{rpar}_0 = g^{b \cdot v_0}, \mathsf{rpk}_0 = Z^{v_0}, \mathsf{rpar}_1 = g^{b \cdot v_1}, \mathsf{rpk}_1 = h$$

where v_0, v_1 are randomly sampled from $\mathbb{Z}_p^*$ and h is randomly sampled from $\mathbb{G}$. If $Z = g^{ab}$, the simulation $(\mathsf{rpar}_0, \mathsf{rpk}_0)$ is indistinguishable from $(\mathsf{rpar}_1, \mathsf{rpk}_1)$, and thus the adversary $\mathcal{A}$ has probability $\frac{1}{2} + \frac{\epsilon}{2}$ of guessing the correct tuple. If $Z \neq g^{ab}$, the adversary only has probability $\frac{1}{2}$ of guessing the correct tuple. So, the advantage of solving the DDH problem is $\frac{\epsilon}{2}$.

4 PPH from Schnorr Identification

In this section, we show how to construct a PPH scheme from identification schemes with map-invariance.

4.1 Generic PPH Construction

Let identification scheme $\mathsf{ID} := (\mathsf{Setup}, \mathsf{IGen}, \mathsf{Com}, \mathsf{Rsp}, \mathsf{Rec}, \mathsf{Apd}, \mathsf{ParMap}, \mathsf{PkMap})$ with system parameter space $\mathcal{SP}$, public key space $\mathcal{PK}$, commitment space $\mathcal{W}$, randomness space $\mathcal{R}$, challenge space $\mathcal{CH}$ and response space $\mathcal{Z}$. Below, we describe the generic PPH construction $\Gamma = (\mathsf{PPH.KeyGen}, \mathsf{PPH.Hash}, \mathsf{PPH.Test})$.

- $\mathsf{PPH.KeyGen}(1^\lambda)$: On input a security parameter λ, it generates $\mathsf{par}_{\mathsf{ID}} \leftarrow \mathsf{Setup}(1^\lambda)$ and two key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$ $(i \in \{0,1\})$ by the key generation algorithm $\mathsf{IGen}(\mathsf{par}_{\mathsf{ID}})$. Define two keyed hash functions $H_r : \{0,1\}^\lambda \times \{0,1\}^* \to \mathcal{R}$ and $H_c : \{0,1\}^\lambda \times \{0,1\}^* \to \mathcal{CH}$, satisfying collision resistance and pseudorandom-function property. Then, it randomly chooses two keys $s, \kappa \leftarrow_\$ \{0,1\}^\lambda$. Finally, it returns the system parameter as $\mathsf{par}_{\mathsf{PPH}} = (\mathsf{par}_{\mathsf{ID}}, H_r, H_c)$, the hash key as $\mathsf{hk} = (s, \kappa, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{pk}_0, \mathsf{pk}_1)$ and the test key as $\mathsf{tk} = \kappa$, where $\mathsf{par}_{\mathsf{PPH}}$ is an implicit input of other algorithms.
- $\mathsf{PPH.Hash}(\mathsf{hk}, u)$: On input the hash key hk, a message u, it computes the values $\hat{u}_0 \leftarrow H_r(s, u)$, $\hat{u}_1 \leftarrow H_r(s, u+1)$ and samples randomness $r_0, r_1 \leftarrow_\$ \mathcal{R}$. Then, it generates the commitments $(\mathsf{cmt}_i, \mathsf{st}_i) \leftarrow \mathsf{Com}(\mathsf{par}_{\mathsf{ID}}; \hat{u}_i \cdot r_i)$, computes the challenge $\mathsf{ch} \leftarrow H_c(\kappa, \hat{u}_0 || \hat{u}_1 || \mathsf{cmt}_0 || \mathsf{cmt}_1)$ and the responses $\mathsf{rsp}_i \leftarrow \mathsf{Rsp}(\mathsf{par}_{\mathsf{ID}}, \mathsf{sk}_i, \mathsf{st}_i, \mathsf{ch})$ for $i \in \{0,1\}$. [6] It computes

$$\mathsf{rpar}_0 \leftarrow \mathsf{ParMap}(\mathsf{par}_{\mathsf{ID}}, r_1), \mathsf{rpk}_0 \leftarrow \mathsf{PkMap}(\mathsf{pk}_0, r_1),$$

$$\mathsf{rpar}_1 \leftarrow \mathsf{ParMap}(\mathsf{par}_{\mathsf{ID}}, r_0), \mathsf{rpk}_1 \leftarrow \mathsf{PkMap}(\mathsf{pk}_1, r_0),$$

  and encodes the challenge by $\mathsf{ech} = \mathsf{ch} \oplus H_c(\kappa, \mathsf{rsp}_0 || \mathsf{rsp}_1)$. Finally, it outputs the hash value $h := \{\mathsf{rpar}_0,\ \mathsf{rpar}_1,\ \mathsf{rpk}_0,\ \mathsf{rpk}_1,\ \mathsf{ech},\ \mathsf{rsp}_0,\ \mathsf{rsp}_1\} \in \mathcal{SP}^2 \times \mathcal{PK}^2 \times \mathcal{CH} \times \mathcal{Z}^2$.
- $\mathsf{PPH.Test}(\mathsf{tk}, h, h')$: On input the test key tk and two hash values

$$h = \{\mathsf{rpar}_0, \mathsf{rpar}_1, \mathsf{rpk}_0, \mathsf{rpk}_1, \mathsf{ech}, \mathsf{rsp}_0, \mathsf{rsp}_1\},$$

$$h' = \{\mathsf{rpar}'_0, \mathsf{rpar}'_1, \mathsf{rpk}'_0, \mathsf{rpk}'_1, \mathsf{ech}', \mathsf{rsp}'_0, \mathsf{rsp}'_1\},$$

[6] Note that $(\mathsf{ch}, \mathsf{rsp}_i)$ is a signature of the message $\hat{u}_0 || \hat{u}_1$ with respect to the public key pk_i, while the secret key is sk_i and the randomness is $\hat{u}_i \cdot r_i$.

it recovers the challenges $\mathsf{ch} = \mathsf{ech} \oplus H_c(\kappa, \mathsf{rsp}_0||\mathsf{rsp}_1)$ and $\mathsf{ch}' = \mathsf{ech}' \oplus H_c(\kappa, \mathsf{rsp}'_0||\mathsf{rsp}'_1)$. Finally, it outputs a flag $b = 1$ if $\mathsf{rec}_0 = \mathsf{rec}'_1$, or $b = -1$ if $\mathsf{rec}_1 = \mathsf{rec}'_0$, otherwise $b = 0$.

$$\begin{cases} \mathsf{rec}_0 \leftarrow \mathsf{Rec}(\mathsf{rpar}'_0, \mathsf{rpk}'_0, \mathsf{ch}, \mathsf{rsp}_0), \\ \mathsf{rec}'_1 \leftarrow \mathsf{Rec}(\mathsf{rpar}_1, \mathsf{rpk}_1, \mathsf{ch}', \mathsf{rsp}'_1), \\ \mathsf{rec}_1 \leftarrow \mathsf{Rec}(\mathsf{rpar}'_1, \mathsf{rpk}'_1, \mathsf{ch}, \mathsf{rsp}_1), \\ \mathsf{rec}'_0 \leftarrow \mathsf{Rec}(\mathsf{rpar}_0, \mathsf{rpk}_0, \mathsf{ch}', \mathsf{rsp}'_0), \end{cases}$$

Correctness. For the predicate $\mathcal{P}$ (in Eq. 6), correctness depends on whether $\mathsf{PPH.Test}(\mathsf{tk}, h, h')$ equals to $\mathcal{P}(x, y)$ for any $x, y \in \mathsf{DSet}$. See Theorem 2 for detailed proof.

Theorem 2. *The proposed PPH scheme Γ is computationally correct under the collision-resistance property of the hash function H_r and the map-invariance property of the identification* ID.

Proof. By the map-invariance property, the identification ID is commitment-augmentable and commitment-independent. So,

$$\begin{cases} \mathsf{rec}_0 = \mathsf{Com}_1(\mathsf{par}_{\mathsf{ID}}; \hat{u}_0 \cdot r_0 \cdot r'_1), \\ \mathsf{rec}_1 = \mathsf{Com}_1(\mathsf{par}_{\mathsf{ID}}; \hat{u}_1 \cdot r_1 \cdot r'_0), \\ \mathsf{rec}'_0 = \mathsf{Com}_1(\mathsf{par}_{\mathsf{ID}}; \hat{u}'_0 \cdot r_1 \cdot r'_0), \\ \mathsf{rec}'_1 = \mathsf{Com}_1(\mathsf{par}_{\mathsf{ID}}; \hat{u}'_1 \cdot r_0 \cdot r'_1), \end{cases}$$

If $\mathsf{rec}_0 = \mathsf{rec}'_1$, then $\hat{u}_0 = \hat{u}'_1$ by the commitment-independency of ID. If a PPT adversary $\mathcal{A}$ finds two values x, y such that both $\mathcal{P}(x, y) \neq 1$ and $\mathsf{rec}_0 = \mathsf{rec}'_1$ hold, $H_r(s, x) = H_r(s, y + 1)$ holds with the state $x \neq y + 1$. Clearly, we can build an adversary $\mathcal{B}$ to break the collision-resistance property of H_r with solution $(x, y + 1)$. So,

$$\Pr\left[\mathcal{P}(x, y) \neq 1 \,|\mathsf{Test}(\mathsf{tk}, h, h') = 1\right] \leq \mathsf{Adv}^{\mathsf{COL}}_{H_r}$$

Similarly, one can prove

$$\Pr\left[\mathcal{P}(x, y) \neq -1 \,|\mathsf{Test}(\mathsf{tk}, h, h') = -1\right] \leq \mathsf{Adv}^{\mathsf{COL}}_{H_r}$$

If $\mathsf{Test}(\mathsf{tk}, h, h') = 0$, it ensures that $H_r(s, x) \neq H_r(s, y + 1)$ and $H_r(s, x) \neq H_r(s, y-1)$, which leads to the fact $x \neq y \pm 1$. Thus, the advantage $\mathsf{Adv}^{\mathsf{COR}}_{\Gamma,\mathcal{P},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{COL}}_{H_r}$ is negligible and the scheme Γ is correct.

4.2 Security Analysis

Theorem 3. *Assume the hash function family H_r is pseudorandom, H_c is entropy smoothing and the identification* ID *is of map-invariance, the proposed PPH scheme Γ is restricted-chosen-input secure.*

Proof. We define a sequence of games as follows:

- Game G_0: This is the real game. Specifically, the challenger generates the hash key hk and tk, samples $h_1 \leftarrow_{\$} \mathsf{HSet}$, and computes challenge hash value $h_0 \leftarrow \mathsf{PPH.Hash}(\mathsf{hk}, x^*)$ where $x^* \in \mathsf{DSet}$ is chosen by $\mathcal{A}$. The challenger picks up a random bit $b \leftarrow_{\$} \{0,1\}$ and sends h_b to $\mathcal{A}$. The adversary $\mathcal{A}$ guesses a bit b'. If $b = b'$, the game outputs 1.
- Game G_1: The pseudorandom function H_r with key s in algorithm $\mathsf{PPH.Hash}$ is replaced by a uniformly random function $f^* : \{0,1\}^* \to \mathcal{R}$.
- Game G_2: The challenge ch in algorithm $\mathsf{PPH.Hash}$ is uniformly sampled from $\mathcal{CH}$.
- Game G_3: The responses $\mathsf{rsp}_0, \mathsf{rsp}_1$ in algorithm $\mathsf{PPH.Hash}$ are uniformly sampled from $\mathcal{Z}^3$.
- Game G_4: $\mathsf{rpar}_0, \mathsf{rpar}_1, \mathsf{rpk}_0, \mathsf{rpk}_1$ in algorithm $\mathsf{PPH.Hash}$ are uniformly sampled from $\mathcal{SP}^2 \times \mathcal{PK}^2$, instead of computing via ParMap and PkMap.

Note that in G_4, all the elements of hash value h_0 are uniformly sampled from appropriate space, and the oracle does not provide any extra information about the bit b. Clearly, the advantage of adversary $\mathcal{A}$ in game G_4 is 0.

Lemma 1. $\mathsf{G}_0 \approx \mathsf{G}_1$ *under the pseudorandom property of the hash function* H_r.

Proof (Proof of Lemma 1). Assume $\mathcal{A}$ is a PPT adversary that can distinguish G_0 and G_1 with non-negligible probability. We can build an adversary $\mathcal{B}$ to break the pseudorandom property of H_r as follows.

$\mathcal{B}$ simulates the PPH security game as prescribed for $\mathcal{A}$, except that the computation of $\hat{u}_0$ and $\hat{u}_1$ in algorithm $\mathsf{PPH.Hash}$ relies on the oracle in the PRF game. When the oracle is $H_r(s,\cdot)$, $\mathcal{B}$ simulates G_0. When the oracle is f^*, $\mathcal{B}$ simulates G_1. So, we have

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{H_r}.$$

Lemma 2. $\mathsf{G}_1 \approx \mathsf{G}_2$ *under the commitment-independency of identification scheme and the entropy smoothing property of hash function* H_c.

Proof (Proof of Lemma 2). Consider game G_1' that is the same as G_1 except that the challenge ch in algorithm $\mathsf{PPH.Hash}$ is computed by $H_c(\kappa, \delta)$ where δ is uniformly sampled from $\mathcal{R}^2 \times \mathcal{W}^2$.

First, we claim that $\mathsf{G}_1 = \mathsf{G}_1'$. In G_1, since $\hat{u}_0$ and $\hat{u}_1$ are generated via random function, these two elements can be viewed as one uniform sampling from $\mathcal{R}^2$. Note that the identification scheme is commitment-independent. In H_1, given fixed system parameters $\mathsf{par}_{\mathsf{ID}}$, the value of cmt_i is determined by randomness $\hat{u}_i \cdot r_i$ for $i \in \{0,1\}$. Since $\hat{u}_i \cdot r_i$ is uniformly distributed over $\mathcal{R}$, cmt_i is uniformly distributed over $\mathcal{W}$.

Then, we claim that $\mathsf{G}_1' \approx \mathsf{G}_2$. Let Q denote the number of hash value queries by adversary $\mathcal{A}$. W.l.o.g., we assume every query made by adversary $\mathcal{A}$ is valid so that the oracle would not return $\perp$. Thus, the oracle would invoke the algorithm $\mathsf{PPH.Hash}$ Q times in G_1'.

Let $\mathsf{G}_{1,0}'$ be the same as G_1' except that challenge ch^* in h_0 is randomly sampled from $\mathcal{CH}$. For $i \in \{1, \cdots, Q\}$, let $\mathsf{G}_{1,i}'$ be the same as $\mathsf{G}_{1,i-1}'$ except that challenge ch in the hash value of the i-th query is randomly sampled from $\mathcal{CH}$. Clearly, $\mathsf{G}'_{1,Q} = \mathsf{G}_2$.

Assume $\mathcal{A}$ is a PPT adversary that can distinguish G'_1 and $\mathsf{G}'_{1,0}$ with non-negligible probability. We can build an adversary $\mathcal{B}$ to break the entropy smoothing property of H_c as follows. $\mathcal{B}$ receives (k, g) from the challenger in the game of entropy smoothing, and sets $\kappa = k$ and $\mathsf{ch}^* = g$. The rest of the simulation is the same as G'_1. When $g = H_c(k, \delta)$ and $\delta \leftarrow_\$ \mathcal{R}^2 \times \mathcal{W}^2$, $\mathcal{B}$ simulates G'_1. When $g \leftarrow_\$ \mathcal{CH}$, $\mathcal{B}$ simulates $\mathsf{G}'_{1,0}$. Similarly, one can prove $\mathsf{G}'_{1,i-1} \approx \mathsf{G}'_{1,i}$ for $i \in \{1, \cdots, Q\}$.

Lemma 3. $\mathsf{G}_2 \approx \mathsf{G}_3$ *under the indistinguishability of response.*

Proof (Proof of Lemma 3). Let game G'_2 be the same as G_2 except that response rsp_0 in algorithm $\mathsf{PPH.Hash}$ is randomly sampled from $\mathcal{Z}$. Let Q be the number of hash value queries by adversary $\mathcal{A}$. Let $\mathsf{G}_{2,0}$ be the same as G_2 except that response rsp_0^* in h_0 is randomly sampled from $\mathcal{Z}$. For $i \in \{1, \cdots, Q\}$, let $\mathsf{G}_{2,i}$ be the same as $\mathsf{G}_{2,i-1}$ except that response rsp_0 in the hash value of the i-th query is randomly sampled from $\mathcal{Z}$. Clearly, $\mathsf{G}_{2,Q} = \mathsf{G}'_2$.

We claim that $\mathsf{G}_2 \approx \mathsf{G}_{2,0}$. Assume that $\mathcal{A}$ can distinguish G_2 and $\mathsf{G}_{2,0}$ with non-negligible probability, we can build an adversary $\mathcal{B}$ to break the indistinguishability of response as follows. The adversary $\mathcal{B}$ receives par, sk, ch and rsp_b from challenger in game $\mathsf{IND}_{\mathsf{ID}}^{\mathsf{RSP}}$, and uses them to simulate game G_2 for $\mathcal{A}$. In particular, $\mathsf{par}_{\mathsf{ID}}$ and sk_0 in algorithm $\mathsf{PPH.KeyGen}$ are set as par and sk respectively. When computing hash value h_0, challenge ch^* is set as ch and rsp_0^* is set as rsp_b. If rsp_b is computed via algorithm Rsp, $\mathcal{B}$ simulates game G_2. Otherwise, rsp_b is randomly sampled from $\mathcal{Z}$ and $\mathcal{B}$ simulates game $\mathsf{G}_{2,0}$. Similarly, one can prove $\mathsf{G}_{2,i-1} \approx \mathsf{G}_{2,i}$ for $i \in \{1, \cdots, Q\}$.

Let game G''_2 be the same as G'_2 except that response rsp_1 in algorithm $\mathsf{PPH.Hash}$ is randomly sampled from $\mathcal{Z}$. Note that G_3 is the same as G''_2 except that response rsp_2 in algorithm $\mathsf{PPH.Hash}$ is randomly sampled from $\mathcal{Z}$. Using same strategy, one can prove $\mathsf{G}'_2 \approx \mathsf{G}''_2$ and $\mathsf{G}''_2 \approx \mathsf{G}_3$.

Lemma 4. $\mathsf{G}_3 \approx \mathsf{G}_4$ *under the map-invariance of identification scheme.*

Proof (Proof of Lemma 4). Let game G'_3 be the same as G_3 except that rpar_0 and rpk_0 in algorithm $\mathsf{PPH.Hash}$ are uniformly sampled from $\mathcal{SP} \times \mathcal{PK}$. Let Q be the number of hash value queries by adversary $\mathcal{A}$. Let $\mathsf{G}_{3,0}$ be the same as G_3 except that rpar_0^* and rpk_0^* in h_0 are uniformly sampled from $\mathcal{SP} \times \mathcal{PK}$. For $i \in \{1, \cdots, Q\}$, let $\mathsf{G}_{3,i}$ be the same as $\mathsf{G}_{3,i-1}$ except that rpar_0 and rpk_0 in the hash value of the i-th query are randomly sampled from $\mathcal{SP} \times \mathcal{PK}$. Clearly, $\mathsf{G}_{3,Q} = \mathsf{G}'_3$.

We claim that $\mathsf{G}_3 \approx \mathsf{G}_{3,0}$. If there exists a PPT adversary $\mathcal{A}$ that can distinguish G_3 and $\mathsf{G}_{3,0}$ with overwhelming probability, we can build a PPT adversary $\mathcal{B}$ to distinguish the distributions of rpar_0^* and rpk_0^* in G_3 and $\mathsf{G}_{3,0}$.

The adversary $\mathcal{B}$ receives an instance of $(\mathsf{par}_{\mathsf{ID}}, \mathsf{rpar}, \mathsf{rpk})$ and simulates the game G_3 as prescribed for $\mathcal{A}$ using $\mathsf{par}_{\mathsf{ID}}$. In particular, $\mathcal{B}$ sets $\mathsf{rpar}_0^* = \mathsf{rpar}$ and $\mathsf{rpk}_0^* = \mathsf{rpk}$. If rpar and rpk are computed via algorithms ParMap and PkMap,

$\mathcal{B}$ simulates G_3. If rpar and rpk are randomly sampled from $\mathcal{SP}$ and $\mathcal{PK}$, $\mathcal{B}$ simulates $\mathsf{G}_{3,0}$. Simlarly, one can prove $\mathsf{G}_{3,i-1} \approx \mathsf{G}_{3,i}$ for $i \in \{1, \cdots, Q\}$.

Let game G_3'' be the same as G_3' except that rpar_1 and rpk_1 in algorithm $\mathsf{PPH.Hash}$ are uniformly sampled from $\mathcal{SP} \times \mathcal{PK}$. Clearly, $\mathsf{G}_3'' = \mathsf{G}_4$ Similarly, one can prove that $\mathsf{G}_3' \approx \mathsf{G}_3''$ under the map-invariance of identification scheme.

To sum up, the advantage of $\mathcal{A}$ is

$$\mathsf{Adv}^{\mathsf{PPH}}_{\Gamma,\mathcal{P},\mathcal{A}} \leq \mathsf{Adv}^{\mathsf{PRF}}_{H_r} + (Q+1)(\mathsf{Adv}^{\mathsf{ES}}_{H_c} + 3(\mathsf{Adv}^{\mathsf{RSP}}_{\mathsf{ID}} + \mathsf{Adv}^{\mathsf{MI}}_{\mathsf{ID}})).$$

Thus, the proposed PPH scheme Γ is restricted-chosen-input secure.

4.3 PPH Instance from Schnorr Identification

The instanced PPH construction Γ based on the Schnorr identification ID with commitment space $\mathbb{G}$, randomness space $\mathbb{Z}_p^*$ and challenge space $\mathbb{Z}_p$ is described formally in Figure 4.

5 The Proposed Parameter-Hiding ORE

In this part, based on our proposed generic PPH schemes from identification schemes, we optimize Cash et al.'s scheme by eliminating bilinear mappings. Finally, we follow the security analysis of previous works [10] to prove the security of our scheme.

5.1 From PPH to Parameter-Hiding ORE

Below, we follow Cash et al's framework [10] to construct a generic parameter-hiding ORE scheme. The first step is to construct ORE with smoothed CLWW leakage based on PPH. Here, consider the same leakage profile $\mathcal{L}_1$ (see Equation 4) as [10]. By definition, it is clear that the ORE construction only leaks the order of underlying plaintexts and the statement whether $\mathsf{msdb}(m_i, m_j)$ and $\mathsf{msdb}(m_i, m_k)$ are the same.

Let $\Gamma = (\mathsf{PPH.KeyGen}, \mathsf{PPH.Hash}, \mathsf{PPH.Test})$ be a PPH scheme with respect to the predicate $\mathcal{P}$, the generic ORE construction is given as follows:

– $\mathsf{ORE.KGen}(1^\lambda)$: On input a security parameter λ, it runs $\mathsf{PPH.KeyGen}(1^\lambda)$ to generate $(\mathsf{par}_{\mathsf{PPH}}, \mathsf{hk}, \mathsf{tk})$. Let $F : \mathcal{K} \times ([n] \times \{0,1\}^n) \to \{0,1\}^\lambda$ be a pseudorandom hash function family. It picks $k \leftarrow_\$ \mathcal{K}$. Define the encoding function $\mathcal{E}(k, m, i) = F(k, i||m_{[:i-1]}||0^{n-i+1}) + m_{[i]} \in \{0,1\}^\lambda$. Then, it outputs the master secret key $\mathsf{msk} = (\mathsf{hk}, k)$ and the comparison key $\mathsf{ck} = \mathsf{tk}$. For brevity, the system parameters $\mathsf{par} = \{\mathsf{par}_{\mathsf{PPH}}, \mathcal{E}\}$ are implicit inputs of other algorithms.

```
PPH.KeyGen($1^\lambda$)
 1: Pick keyed hash functions $H_r : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p^*$ and $H_c : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p$.
 2: $g \leftarrow_\$ \mathbb{G}$, $s, \kappa, x_0, x_1 \leftarrow_\$ \mathbb{Z}_p$
 3: $y_0 := g^{x_0}$, $y_1 := g^{x_1}$
 4: $\mathsf{par}_{\mathsf{PPH}} := \{g, H_r, H_c\}$
 5: $\mathsf{hk} := \{s, \kappa, x_0, x_1, g, y_0, y_1\}$, $\mathsf{tk} := \{\kappa\}$
 6: return $(\mathsf{par}_{\mathsf{PPH}}, \mathsf{hk}, \mathsf{tk})$

PPH.Hash(hk, $u$)
 1: $\{s, \kappa, x_0, x_1, g, y_0, y_1\} \leftarrow \mathsf{hk}$
 2: $\hat{u}_0 \leftarrow H_r(s, u)$, $\hat{u}_1 \leftarrow H_r(s, u+1)$
 3: $r_0, r_1 \leftarrow_\$ \mathbb{Z}_p$, $w_0 := g^{\hat{u}_0 \cdot r_0}$, $w_1 := g^{\hat{u}_1 \cdot r_1}$
 4: $\xi \leftarrow H_c(\hat{u}_0 || \hat{u}_1 || w_0 || w_1)$
 5: $z_0 := \hat{u}_0 \cdot r_0 - \xi \cdot x_0$, $z_1 := \hat{u}_1 \cdot r_1 - \xi \cdot x_1$
 6: $\hat{g}_0 := g^{r_1}$, $\hat{y}_0 := y_0^{r_1}$, $\hat{g}_1 := g^{r_0}$, $\hat{y}_1 := y_1^{r_0}$
 7: $\hat{\xi} := \xi \oplus H_c(\kappa || z_0 || z_1)$
 8: return $H := \{\hat{g}_0, \hat{g}_1, \hat{y}_0, \hat{y}_1, \hat{\xi}, z_0, z_1\}$

PPH.Test(tk, $h, h'$)
 1: $\{\hat{g}_0, \hat{g}_1, \hat{y}_0, \hat{y}_1, \hat{\xi}, z_0, z_1\} \leftarrow h$
 2: $\{\hat{g}'_0, \hat{g}'_1, \hat{y}'_0, \hat{y}'_1, \hat{\xi}', z'_0, z'_1\} \leftarrow h'$
 3: $\xi := \hat{\xi} \oplus H_c(\kappa || z_0 || z_1)$, $\xi' := \hat{\xi}' \oplus H_c(\kappa || z'_0 || z'_1)$
 4: $\mathsf{rec}_0 := (\hat{g}'_0)^{z_0} \cdot (\hat{y}'_0)^{\xi}$, $\mathsf{rec}'_1 := (\hat{g}_1)^{z'_1} \cdot (\hat{y}_1)^{\xi'}$
 5: if $\mathsf{rec}_0 = \mathsf{rec}'_1$ then
 6:     return 1
 7: $\mathsf{rec}_1 := (\hat{g}'_1)^{z_1} \cdot (\hat{y}'_1)^{\xi}$, $\mathsf{rec}'_0 := (\hat{g}_0)^{z'_0} \cdot (\hat{y}_0)^{\xi'}$
 8: if $\mathsf{rec}_1 = \mathsf{rec}'_0$ then
 9:     return $-1$
10: return 0
```

Fig. 4. The instanced PPH construction based on the Schnorr identification ID.

– ORE.Enc(msk, m): On input msk and a n bits message m, it randomly selects a permutation $\pi : [n] \to [n]$ and computes

$$\begin{cases} u_i = \mathcal{E}(k, m, \pi(i)), \\ h_i \leftarrow \mathsf{PPH.Hash}(\mathsf{hk}, u_i), \end{cases} \forall i \in [n]$$

Finally, it outputs the ciphertext

$$\vec{c} := \{h_1, \cdots, h_n\} \in \mathcal{SP}^{2n} \times \mathcal{PK}^{2n} \times \mathcal{CH}^n \times \mathcal{Z}^{2n}.$$

– ORE.Cmp(ck, $\vec{c}, \vec{c}'$): On input the comparison key ck and two ciphertexts $(\vec{c}, \vec{c}')$, it computes

$$b_{ij} \leftarrow \mathsf{PPH.Test}(\mathsf{ck}, h_i, h'_j),\ \forall i, j \in [n]$$

and stops when $b_{ij} \neq 0$. If it stops with $b_{ij} = 1$, it outputs 1 to indicate that $m > m'$; else if it stops with $b_{ij} = -1$, it outputs -1 to indicate that $m > m'$. If there does not exist i, j such that $b_{ij} \neq 0$, it outputs 0 to indicate that $m = m'$.

The above description differs from the original construction [10] mainly in that the predicate corresponding to PPH adds an additional case, i.e $\mathcal{P}(x, y) = -1$ if and only if $x < y$. In the original construction, the predicate is defined as $\mathcal{P}(x, y) = 1$ if and only if $x > y$, which results in that it needs to execute $\mathsf{PPH.Test}$ algorithm twice for each c_i, c'_j, i.e. $\mathsf{PPH.Test}(\mathsf{ck}, h_i, h'_j)$ and $\mathsf{PPH.Test}(\mathsf{ck}, h'_j, h_i)$. But, our generic PPH construction only needs to execute $\mathsf{PPH.Test}$ algorithm once for each h_i, h'_j.

Correctness. For the predicate $\mathcal{O}$ (in Eq. 1), correctness depends on whether $\mathsf{ORE.Cmp}(\mathsf{ck}, c, c')$ equals to $\mathcal{O}(m, m')$ for any $m, m' \in \mathcal{D}$. See Theorem 4 for detailed proof.

Theorem 4. *The proposed ORE scheme Π is computationally correct under the pseudorandom property of the hash function F and the computational correctness of the PPH scheme Γ.*

Proof. If the PPH scheme Γ is computationally correct, b_{ij} equals to $\mathcal{P}(u_i, u'_j)$ with overwhelming probability. Considering the first case $\mathcal{O}(m, m') = 1$, there must be two indexes (i, j) satisfying $\pi(i) = \pi'(j')$ and $\mathcal{P}(u_i, u'_j) = 1$, since the function F with the key s outputs deterministic results. Here,

$$\begin{aligned}
&\mathcal{E}(k, m, \pi(i)) = \mathcal{E}(k, m', \pi'(j)) + 1 \Rightarrow \\
&F(k, \pi(i)||m_{[:\pi(i)-1]}||0^{n-\pi(i)+1}) + m_{[\pi(i)]} \\
=&F(k, \pi'(j)||m'_{[:\pi'(j)-1]}||0^{n-\pi'(j)+1}) + m'_{[\pi'(j)]} + 1.
\end{aligned}$$

As long as there is no indexes (i', j') satisfying $\mathcal{P}(u_{i'}, u'_{j'}) = -1$, $\mathsf{ORE.Cmp}(\mathsf{ck}, c, c')$ must output 1. For all $(i', j') \neq (i, j)$, we have

$$\Pr\left[\mathcal{P}(u_{i'}, u'_{j'}) = -1\right] \leq 1 - \left(1 - \frac{n}{2^\lambda}\right)^n$$

In this case, the probability of $\mathsf{ORE.Cmp}$ $(\mathsf{ck}, c, c') \neq \mathcal{O}(m, m')$ is negligible. Similarly, one can prove this probability is also negligible in the case $\mathcal{O}(m, m') = -1$ and $\mathcal{O}(m, m') = 0$. Thus, the proposed ORE scheme Π is correct.

Security. The proof of Theorem 5 is similar to Theorem 12 in [10], so we omit the details here.

Theorem 5. *Given a secure pseudorandom function F, if the underlying PPH scheme Γ is restricted-chosen-input secure, the generic ORE scheme is $\mathcal{L}_1$-non-adaptive-simulation secure.*

Starting from Cash et al.'s framework, one can easily construct pORE schemes from identification-based PPH schemes. For instance with identification schemes, the hash key and the test key are modified by $\mathsf{hk} := \{s, \kappa, \mathsf{sk}_0, \mathsf{sk}_1, \mathsf{pk}_0, \mathsf{pk}_1\}$ and $\mathsf{tk} := \{\kappa\}$, respectively. For each $i \in [n]$, the hash value h_i is composed of $\{\mathsf{rpar}_{i,0}$, $\mathsf{rpar}_{i,1}$, $\mathsf{rpk}_{i,0}$, $\mathsf{rpk}_{i,1}$, ech_i, $\mathsf{rsp}_{i,0}$, $\mathsf{rsp}_{i,1}\}$. Clearly, the ciphertext $\vec{c}$ belongs to the space $\mathcal{SP}^{2n} \times \mathcal{PK}^{2n} \times \mathcal{CH}^n \times \mathcal{Z}^{2n}$ and the comparison algorithm Cmp needs to perform n^2 times Hash algorithms, equivalent to $4n^2$ times Rec operations.

5.2 ORE Instance from Schnorr Identification

Now, from Schnorr identification, we construct an efficient ORE scheme, named $\mathsf{Sch\text{-}ORE}$, without bilinear pairings in Figure 5. The master secret key consists of two randomness $s, \kappa \leftarrow_\$ \mathbb{Z}_p$, two secret keys $x_0, x_1 \leftarrow_\$ \mathbb{Z}_p$ and public keys $\{g, y_0, y_1\} \in \mathbb{G}$, while the comparison key is only the randomness κ. For each bit-wise hash value h_i, its size is $4|\mathbb{G}| + 3|\mathbb{Z}_p|$. For the comparison algorithm, at most $8n^2$ group exponentiation operations are required to compare two ciphertexts with respect to any messages. Essentially, our ORE scheme uses two group exponentiation operations to replace one bilinear pairing operation. Theoretically, the comparison efficiency can be improved by about 3 to 4 times with the same security level.

Improving Efficiency with $\mathcal{L}'_1$ Leakage. Cash et al. [10] proposed the optimization idea of fixing the permutation π in msk to replace randomized permutation in $\mathsf{PPH.Hash}$ algorithm. The optimization point is that it only executes the $\mathsf{PPH.Test}$ algorithm on the same index, i.e. $i = j$, in the comparison algorithm. Combined with our optimization points, each comparison algorithm requires $8n$ times group exponentiation operations, but it can lead to a weaker leakage $\mathcal{L}'_1$ (in Equation 5). Compared to $\mathcal{L}_1$ (in Equation 4), $\mathcal{L}'_1$ reveals extra information that $\mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_k, m_l))$ even when $i \neq k$. According to Theorem 12 in [10], it can still be confirmed that our $\mathsf{Sch\text{-}ORE}$ is $\mathcal{L}'_1$-non-adaptive-simulation secure if Γ is restricted-chosen-input secure and F is a secure pseudorandom function.

6 Experimental Evaluation

In order to evaluate the performance of our parameter-hiding ORE scheme, we built and evaluated an implementation with instantiated components [7]. Then, we describe the comparison results with Cash et al.'s ORE scheme [10] in the aspect of storage and computation costs under different plaintext bit-lengths.

Instantiating Primitives. Our implementation is written in C language. We instantiate the necessary keyed hash function using SHA-256. We use GMP library to implement multi-precision integer arithmetic and PBC library to implement bilinear pairings. Notice that there is no need to use bilinear pairing operation in our schemes, but to ensure fairness, we uniformly use the $\mathbb{G}_1$ group

[7] Our implementation is available at https://github.com/cpeng-crypto/pORE.

```
$\underline{\mathsf{ORE.KGen}(1^\lambda)}$
1: Pick keyed hash functions $H_r : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p^*$ and $H_c : \{0,1\}^\lambda \times \{0,1\}^* \to \mathbb{Z}_p$.
2: $g \leftarrow_\$ \mathbb{G}$, $s, \kappa, x_0, x_1 \leftarrow_\$ \mathbb{Z}_p$
3: $y_0 := g^{x_0}$, $y_1 := g^{x_1}$
4: $\mathsf{par}_{\mathsf{PPH}} := \{g, H_r, H_c\}$
5: $\mathsf{hk} := \{s, \kappa, x_0, x_1, g, y_0, y_1\}$, $\mathsf{tk} := \{\kappa\}$
6: $k \leftarrow_\$ \mathcal{K}$
7: $\mathcal{E}(k, m, i) := F(k, i||m_{[:i-1]}||0^{n-i+1}) + m_{[i]}$
8: $\mathsf{par} := (\mathsf{par}_{\mathsf{PPH}}, \mathcal{E})$, $\mathsf{msk} := (\mathsf{hk}, k)$, $\mathsf{ck} := \mathsf{tk}$
9: return $(\mathsf{par}, \mathsf{msk}, \mathsf{ck})$
$\underline{\mathsf{ORE.Enc}(\mathsf{msk}, m)}$
1: $\{s, \kappa, x_0, x_1, g, y_0, y_1, k\} \leftarrow \mathsf{msk}$
2: $\pi : [n] \to [n]$
3: for $i \in [n]$ do
4:     $r_{i,0}, r_{i,1} \leftarrow_\$ \mathbb{Z}_p^*$
5:     $u_i := \mathcal{E}(k, m, \pi(i))$
6:     $\hat{u}_{i,0} \leftarrow H_r(s, u_i)$, $\hat{u}_{i,1} \leftarrow H_r(s, u_i + 1)$
7:     $w_{i,0} := g^{\hat{u}_{i,0} \cdot r_{i,0}}$, $w_{i,1} := g^{\hat{u}_{i,1} \cdot r_{i,1}}$
8:     $\xi_i \leftarrow H_c(\hat{u}_{i,0}||w_{i,0}||\hat{u}_{i,1}||w_{i,1})$
9:     $z_{i,0} := \hat{u}_{i,0} \cdot r_{i,0} - \xi_i \cdot x_0$
10:    $z_{i,1} := \hat{u}_{i,1} \cdot r_{i,1} - \xi_i \cdot x_1$
11:    $\hat{g}_{i,0} := g^{r_{i,1}}$, $\hat{y}_{i,0} := y_0^{r_{i,1}}$, $\hat{g}_{i,1} := g^{r_{i,0}}$, $\hat{y}_{i,1} := y_1^{r_{i,0}}$
12:    $\hat{\xi}_i := \xi_i \oplus H_c(\kappa||z_{i,0}||z_{i,1})$
13:    $h_i := \{\hat{g}_{i,0}, \hat{y}_{i,0}, \hat{g}_{i,1}, \hat{y}_{i,1}, \hat{\xi}_i, z_{i,0}, z_{i,1}\}$
14: $\vec{c} := \{h_1, \cdots, h_n\}$;
15: return $\vec{c}$
$\underline{\mathsf{ORE.Cmp}(\mathsf{ck}, \vec{c}, \vec{c}')}$
1: $\{h_1, \cdots, h_n\} \leftarrow \vec{c}$, $\{h'_1, \cdots, h'_n\} \leftarrow \vec{c}'$
2: $\{\hat{g}_{i,0}, \hat{y}_{i,0}, \hat{g}_{i,1}, \hat{y}_{i,1}, \hat{\xi}_i, z_{i,0}, z_{i,1}\} \leftarrow h_i$
3: $\{\hat{g}'_{j,0}, \hat{y}'_{j,0}, \hat{g}'_{j,1}, \hat{y}'_{j,1}, \hat{\xi}'_j, z'_{j,0}, z'_{j,1}\} \leftarrow h'_j$
4: for $i \in [n], j \in [n]$ do
5:     $\xi_i := \hat{\xi}_i \oplus H_c(\kappa||z_{i,0}||z_{i,1})$
6:     $\xi'_j := \hat{\xi}'_j \oplus H_c(\kappa||z'_{j,0}||z'_{j,1})$
7:     $\mathsf{rec}_{i,j,0} := (\hat{g}'_{j,0})^{z_{i,0}} \cdot (\hat{y}'_{j,0})^{\xi_i}$, $\mathsf{rec}'_{i,j,1} := (\hat{g}_{i,1})^{z'_{j,1}} \cdot (\hat{y}_{i,1})^{\xi'_j}$
8:     if $\mathsf{rec}_{i,j,0} = \mathsf{rec}'_{i,j,1}$ then                ▷ check $u_i = u'_j + 1$?
9:         return $b_{ij} = 1$
10:    $\mathsf{rec}_{i,j,1} = (\hat{g}'_{j,1})^{z_{i,1}} \cdot (\hat{y}'_{j,1})^{\xi_i}$, $\mathsf{rec}'_{i,j,0} = (\hat{g}_{i,0})^{z'_{j,0}} \cdot (\hat{y}_{i,0})^{\xi'_j}$
11:    if $\mathsf{rec}_{i,j,1} = \mathsf{rec}'_{i,j,0}$ then                ▷ check $u_i + 1 = u'_j$?
12:        return $b_{ij} = -1$
13: return $b = 0$
```

Fig. 5. Sch-ORE scheme, the instanced ORE construction based on the Schnorr identification.

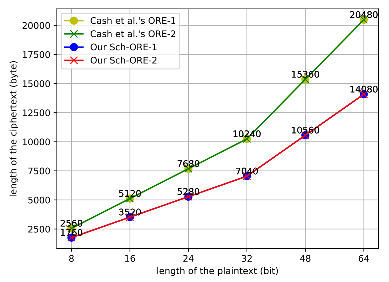
n	Scheme	Ciphertext size (KB)	Encryption cost (ms)	Comparison Cost (ms)	Leakage
8	Cash et al.'s ORE [10]	2.50	43.52	487.17	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	2.50	43.52	60.91	$\mathcal{L}_1'$
	Our Sch-ORE	1.72	16.71	151.06	$\mathcal{L}_1$
	Our Sch-ORE	**1.72**	**16.71**	**19.81**	$\mathcal{L}_1'$
16	Cash et al.'s ORE [10]	5.00	87.04	1948.67	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	5.00	87.04	121.97	$\mathcal{L}_1'$
	Our Sch-ORE	3.44	33.42	602.14	$\mathcal{L}_1$
	Our Sch-ORE	**3.44**	**33.42**	**39.62**	$\mathcal{L}_1'$
24	Cash et al.'s ORE [10]	7.50	130.56	4384.51	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	7.50	130.56	182.69	$\mathcal{L}_1'$
	Our Sch-ORE	5.16	50.12	1353.23	$\mathcal{L}_1$
	Our Sch-ORE	**5.16**	**50.12**	**59.43**	$\mathcal{L}_1'$
32	Cash et al.'s ORE [10]	10.00	174.08	7794.69	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	10.00	174.08	243.58	$\mathcal{L}_1'$
	Our Sch-ORE	6.88	66.83	2404.32	$\mathcal{L}_1$
	Our Sch-ORE	**6.88**	**66.83**	**79.24**	$\mathcal{L}_1'$
48	Cash et al.'s ORE [10]	15.00	261.12	17538.05	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	15.00	261.12	365.38	$\mathcal{L}_1'$
	Our Sch-ORE	10.31	100.25	5406.56	$\mathcal{L}_1$
	Our Sch-ORE	**10.31**	**100.25**	**118.85**	$\mathcal{L}_1'$
64	Cash et al.'s ORE [10]	20.00	348.16	31178.75	$\mathcal{L}_1$
	Cash et al.'s ORE* [10]	20.00	348.16	487.17	$\mathcal{L}_1'$
	Our Sch-ORE	13.75	133.67	9608.83	$\mathcal{L}_1$
	Our Sch-ORE	**13.75**	**133.67**	**158.47**	$\mathcal{L}_1'$

Table 2. Ciphertext size in KB and running time in milliseconds of ORE schemes with different message bit-length n in $\{8, 16, 24, 32, 48, 64\}$.ORE* refers to the ORE scheme with fixed permutation.

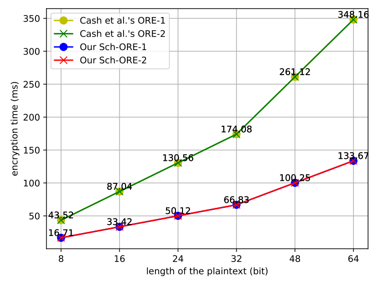
of bilinear pairing as the cyclic group $\mathbb{G}$ in our schemes. We believe this provides a more balanced comparison of the performance tradeoffs between Cash et al.'s scheme and our new scheme.

Security Parameters. All evaluations were performed with the bilinear pairing parameter set "*d159.param*" under the security parameter $\lambda = 80$ bits. In such case, Symmetric eXternal Diffie-Hellman (SXDH) assumption holds, which is the basis for the security of parameter-hiding ORE in [10]. Specifically, the size of elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ is $|\mathbb{G}_1| = 40$ bytes and $|\mathbb{G}_2| = 120$ bytes, respectively.

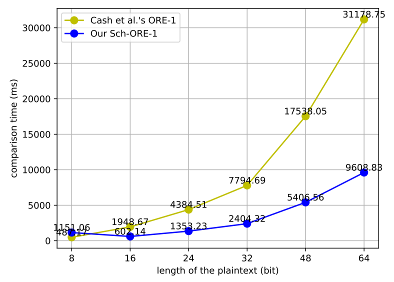
The size of elements in $\mathbb{Z}_q^*$ is $|\mathbb{Z}_q^*| = 20$ bytes. Note that elements in group $\mathbb{G}_2$ need only be 3 times longer than elements in group $\mathbb{G}_1$.



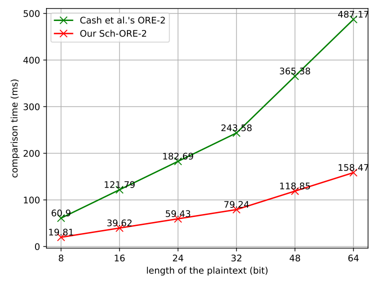
(a) Ciphertext size in ORE



(b) Encryption time in ORE



(c) Comparison time in ORE with randomized permutations



(d) Comparison time in ORE with fixed permutations

Fig. 6. Performance comparison

Benchmarks and Evaluation. All experiments are executed on a desktop with Intel(R) Core(TM) i5-10500 CPU @ 3.10GHz, 16GB RAM and Ubuntu 18.04 LTS. Although both Cash et al.'s scheme and our scheme are easily parallelizable, we do not exploit parallelism in our benchmarks. We report detailed comparisons in Table 2 for different plaintext bit-length $n \in \{8, 16, 24, 32, 48, 64\}$. As expected, our protocol shows a significant performance improvement for each bit-length. For a more intuitive presentation, we divide into these results into three categories for comparison.

Ciphertext Size. As shown in Figure 6(a), ciphertexts encrypted with random permutation or fixed permutation have the same bit length for the same plaintext length. For n bits plaintext, our Sch-ORE scheme costs $4n|\mathbb{G}_1| + 3n|\mathbb{Z}_p| \approx 220n$ bytes. Compared to Cash et al.'s ORE [10], our Sch-ORE scheme can reduce the ciphertext size by about 31.25%. However, the ciphertext expansion ratio is still large, for example, the ciphertext size reaches 13.75 KB at 64-bit plaintext length.

Encryption Efficiency. For encrypted n-bit messages, roughly $6n$ $\mathbb{G}_1$ group exponentiation operations need to be computed in Sch-ORE scheme while $2n$ $\mathbb{G}_1$ group exponentiation and $2n$ $\mathbb{G}_2$ group exponentiation operations in [10]. In contrast, $\mathbb{G}_2$ is an elliptic curve point group over $\mathbb{F}_{p^3}$, which is nearly 7 times less efficient in terms of point multiplication than the elliptic curve point group $\mathbb{G}_2$ over $\mathbb{F}_p$. As shown in Figure 6(b), for the whole encryption algorithm, our scheme is almost 2.6 times faster than Cash et al.'s ORE [10].

Comparison Efficiency. In Fig.6(c) and 6(d), we evaluate the comparative efficiency of various ORE schemes. It show that in the case of randomized permutations, the comparison time consumption grows rapidly with the increase of plaintext bits, but for fixed permutations, it grows more slowly. Clearly, the comparative efficiency of random permutations is n times slower than that of fixed permutations. Besides, in both modes, our Sch-ORE is more than 3 times faster than ORE in [10]. For 64-bit plaintext, it needs 158 ms for a single comparison.

To sum up, our Sch-ORE scheme performs better than other parameter-hiding ORE at the same security level and leakage, in which the ciphertext length is reduced by more than 31.25%, the encryption efficiency increased by nearly 2.6 times and the comparison efficiency increased by more than 3 times.

7 Conclusion

In this paper, we proposed an efficient and secure construction for parameter-hiding order-revealing encryption (pORE). Our approach relies on the map-invariance property for identification protocols, which enables us to develop a generic construction of property-preserving hash (PPH) schemes. Using our PPH scheme, we presented a new parameter-hiding ORE that outperforms existing state-of-the-art constructions. Specifically, we instantiated our framework with the pairing-free Schnorr identification scheme and demonstrated that our proposed pORE scheme achieves a reduction in ciphertext size of approximately 31.25%, while improving encryption and comparison efficiency by more than two times.

Our work offers a practical and secure alternative to existing pORE constructions. As future work, it is interesting to instant our ID schemes (with map-variance) from other hardness assumptions. For example, it seems like that one could construct ID scheme based on isogeny-related assumptions for post-quantum pORE. Note that it is difficult to extend previous construction for post-quantum security due to the need of pairing computation for comparison, while our work enables more potential for constructing pORE with better security and efficiency.

Acknowlegements. We thank the anonymous reviewers for their helpful discussion and feedback. The work was supported by the National Key Research and Development Program of China (No. 2022YFB3102400), the National Natural Science Foundation of China (Nos. U21A20466, 62325209, 62272350,62122092, 62032005, 62202485), and the Major Program(JD) of Hubei Province (No. 2023BA A027).

References

1. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data. pp. 563–574 (2004)
2. Arasu, A., Blanas, S., Eguro, K., Kaushik, R., Kossmann, D., Ramamurthy, R., Venkatesan, R.: Orthogonal security with cipherbase. In: CIDR (2013)
3. Bajaj, S., Sion, R.: Trusteddb: A trusted hardware-based database with privacy and data confidentiality. IEEE Transactions on Knowledge and Data Engineering **26**(3), 752–765 (2013)
4. Bogatov, D., Kollios, G., Reyzin, L.: A comparative evaluation of order-revealing encryption schemes and secure range-query protocols. Proceedings of the VLDB Endowment **12**(8), 933–947 (2019)
5. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 224–241. Springer, Heidelberg (Apr 2009). https://doi.org/10.1007/978-3-642-01001-9_13
6. Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: Improved security analysis and alternative solutions. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 578–595. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-22792-9_33
7. Boneh, D., Kogan, D., Woo, K.: Oblivious pseudorandom functions from isogenies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 520–550. Springer (2020)
8. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_19
9. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 668–679 (2015)
10. Cash, D., Liu, F.H., O'Neill, A., Zhandry, M., Zhang, C.: Parameter-hiding order revealing encryption. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part I. LNCS, vol. 11272, pp. 181–210. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03326-2_7
11. Chang, Z., Xie, D., Li, F.: Oblivious ram: A dissection and experimental evaluation. Proceedings of the VLDB Endowment **9**(12), 1113–1124 (2016)
12. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical order-revealing encryption with limited leakage. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 474–493. Springer, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-52993-5_24
13. Demertzis, I., Papadopoulos, S., Papapetrou, O., Deligiannakis, A., Garofalakis, M.: Practical private range search revisited. In: Proceedings of the 2016 International Conference on Management of Data. pp. 185–198 (2016)
14. Durak, F.B., DuBuisson, T.M., Cash, D.: What else is revealed by order-revealing encryption? In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1155–1166 (2016)
15. Faber, S., Jarecki, S., Krawczyk, H., Nguyen, Q., Rosu, M.C., Steiner, M.: Rich queries on encrypted data: Beyond exact matches. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) ESORICS 2015, Part II. LNCS, vol. 9327, pp. 123–145. Springer, Heidelberg (Sep 2015). https://doi.org/10.1007/978-3-319-24177-7_7

16. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) **33**(4), 792–807 (1986)
17. Grubbs, P., Lacharité, M.S., Minaud, B., Paterson, K.G.: Pump up the volume: Practical database reconstruction from volume leakage on range queries. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 315–331. ACM Press (Oct 2018). https://doi.org/10.1145/3243734.3243864
18. Grubbs, P., Lacharité, M.S., Minaud, B., Paterson, K.G.: Learning to reconstruct: Statistical learning theory and encrypted database attacks. In: 2019 IEEE Symposium on Security and Privacy. pp. 1067–1083. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00030
19. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., Ristenpart, T.: Leakage-abuse attacks against order-revealing encryption. In: 2017 IEEE Symposium on Security and Privacy. pp. 655–672. IEEE Computer Society Press (May 2017). https://doi.org/10.1109/SP.2017.44
20. Hirose, S.: Collision-resistant and pseudorandom function based on merkle-damgård hash function. In: International Conference on Information Security and Cryptology. pp. 325–338. Springer (2022)
21. Kamara, S., Moataz, T.: Computationally volume-hiding structured encryption. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 183–213. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17656-3_7
22. Kellaris, G., Kollios, G., Nissim, K., O'Neill, A.: Generic attacks on secure outsourced databases. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1329–1340. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978386
23. Kerschbaum, F.: Frequency-hiding order-preserving encryption. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 656–667. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813629
24. Kerschbaum, F., Schröpfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 2014. pp. 275–286. ACM Press (Nov 2014). https://doi.org/10.1145/2660267.2660277
25. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 552–586. Springer (2018)
26. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Annual International Cryptology Conference. pp. 33–61. Springer (2016)
27. Kornaropoulos, E.M., Papamanthou, C., Tamassia, R.: The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution. In: 2020 IEEE Symposium on Security and Privacy. pp. 1223–1240. IEEE Computer Society Press (May 2020). https://doi.org/10.1109/SP40000.2020.00029
28. Kornaropoulos, E.M., Papamanthou, C., Tamassia, R.: Response-hiding encrypted ranges: Revisiting security via parametrized leakage-abuse attacks. In: 2021 IEEE Symposium on Security and Privacy. pp. 1502–1519. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00044
29. Lacharité, M.S., Minaud, B., Paterson, K.G.: Improved reconstruction attacks on encrypted data using range query leakage. In: 2018 IEEE Symposium on Security and Privacy. pp. 297–314. IEEE Computer Society Press (May 2018). https://doi.org/10.1109/SP.2018.00002

30. Lewi, K., Wu, D.J.: Order-revealing encryption: New constructions, applications, and lower bounds. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016. pp. 1167–1178. ACM Press (Oct 2016). https://doi.org/10.1145/2976749.2978376
31. Markatou, E.A., Tamassia, R.: Full database reconstruction with access and search pattern leakage. In: Lin, Z., Papamanthou, C., Polychronakis, M. (eds.) ISC 2019. LNCS, vol. 11723, pp. 25–43. Springer, Heidelberg (Sep 2019). https://doi.org/10.1007/978-3-030-30215-3_2
32. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015. pp. 644–655. ACM Press (Oct 2015). https://doi.org/10.1145/2810103.2813651
33. Pandey, O., Rouselakis, Y.: Property preserving symmetric encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 375–391. Springer (2012)
34. Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. In: 2013 IEEE Symposium on Security and Privacy. pp. 463–477. IEEE Computer Society Press (May 2013). https://doi.org/10.1109/SP.2013.38
35. Popa, R.A., Redfield, C.M., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. pp. 85–100 (2011)
36. Teranishi, I., Yung, M., Malkin, T.: Order-preserving encryption secure beyond one-wayness. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 42–61. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_3

A More on the Leakage of Different ORE Schemes

As shown by Cash et al. [10], $\mathcal{L}_1$ leaks less information than the leakage caused by CLWW ORE ($\mathcal{L}_2$) and Lewi-Wu ORE. Below we will provide more details about these leakage profiles.

Smoothed CLWW Leakage. Considering the set $\{0, 4, 5, 10, 11\}$ in 4-bit plaintext space, it can be viewed as leaves of a 4-level full binary tree. In Figure 7, the ideal leakage $\mathcal{L}_0$ refers to the numeric order-relation. After removing the irrelevant leaves/nodes, the CLWW leakage $\mathcal{L}_2$ refers to the subtree structure, in which grey points mean the position of msdb and green nodes mean unleaked msdb. Note that for some plaintexts such as $\{2, 6, 7, 12, 13\}$, it would also have equivalent subtree w.r.t. $\{0, 4, 5, 10, 11\}$. While for other plaintexts such as $\{1, 2, 3, 5, 6\}$, there are significant differences between the two subtree structures, and these two plaintext sequences are distinguishable by the leakage $\mathcal{L}_2$ of the ciphertext alone. Moreover, the comparator can know from the ciphertext that $m_i = 4$ has the bit form "01-0", where "-" indicates the unknown bits. Considering the leakage $\mathcal{L}_1$ which leaks the equality pattern of msdb, while the comparator can infer additional information (i.e., $\mathsf{msdb}(0,4) < \mathsf{msdb}(5, 11)$), green nodes in subtrees are unknown to the comparator as the positions of msdb are not determined. Also, the position of gray nodes can be moved up or down. So, one can see that $\{0, 4, 5, 10, 11\}$ and $\{1, 2, 3, 5, 6\}$ leak the same information under $\mathcal{L}_1$. Note that inspired by the block-wise encryption [30], Cash et al. [10] also

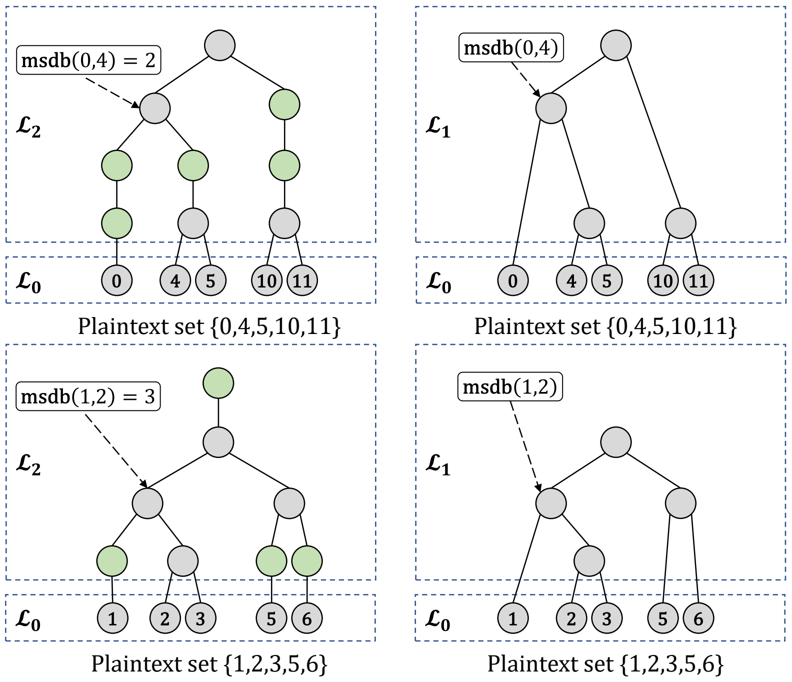


Fig. 7. Comparison of different leakage profiles.

demonstrated that an enhanced level of leakage $\mathcal{L}_1$ can be achieved by encrypting message blocks instead of individual bits.

Specific Applications of pORE. Cash et al. [10] show that pORE is particularly suitable for specific scenarios where the adversary lacks a strong estimate of the prior distribution of the data. In many settings, data often follows a known type of distribution, as exemplified by Cash et al. [10], where various physical, biological, and financial quantities approximate a normal distribution due to the central limit theorem. In these scenarios, the database entries are independently drawn from a distribution with a known “shape" (e.g., normal, uniform, Laplace, etc.), but the adversary does not possess the mean and variance information necessary to determine the shifting and scaling factors. Consequently, pORE can be employed to effectively conceal both the shifting and scaling information. Notably, it has been observed by Cash et al. [10] that CLWW ORE [12] and Lewis-Wu ORE [30] fail to achieve shift hiding and scale hiding simultaneously. Nonetheless, as acknowledged by Cash et al. [10], pORE specifically guarantees security when the sensitivity lies solely in the scale and shift of the underlying plaintext distributions, and it may not be sufficient in scenarios where the shape of the distribution itself is highly sensitive or when there are correlations with other available data that could be exploited by an attacker.