Parameter-Hiding Order Revealing Encryption

David Cash[1], Feng-Hao Liu[2], Adam O'Neill[3], Mark Zhandry[4], Cong Zhang[5]

[1] Department of Computer Science, University of Chicago
[2] Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University
[3] Department of Computer Science, Georgetown University
[4] Department of Computer Science, Princeton University
[5] Department of Computer Science, Rutgers University

Abstract. Order-revealing encryption (ORE) is a primitive for outsourcing encrypted databases which allows for efficiently performing range queries over encrypted data. Unfortunately, a series of works, starting with Naveed *et al.* (CCS 2015), have shown that when the adversary has a good estimate of the distribution of the data, ORE provides little protection. In this work, we consider the case that the database entries are drawn identically and independently from a distribution of known shape, but for which the mean and variance are not (and thus the attacks of Naveed *et al.* do not apply). We define a new notion of security for ORE, called *parameter-hiding ORE*, which maintains the secrecy of these parameters. We give a construction of ORE satisfying our new definition from bilinear maps.

Keywords. Encryption, Order-revealing encryption.

1 Introduction

An emerging area of cryptography concerns the design and analysis of "leaky" protocols (see *e.g.* [33, 36, 11] and additional references below), which are protocols that deliberately give up some level of security in order to achieve better efficiency. One important tool in this area is *order-revealing encryption* [7, 8][6]. Order-revealing encryption (ORE) is a special type of symmetric encryption which leaks the order of the underlying plaintexts through a *public* procedure Comp. In practice, ORE allows for a client to store a database on an untrusted server in encrypted form, while still permitting the server to efficiently perform various operations such as range queries on the encrypted data without the secret decryption key. ORE has been implemented and used in real-world encrypted database systems, including CryptDB [36].

Various notions of ORE have been proposed. The strongest, called "ideal" ORE, insists that everything about the plaintexts is hidden, except for their order. For example, it should be impossible to distinguish between encryptions of $1, 2, 3$ and $1, 4, 9$. Such ideal ORE can be constructed from multilinear maps [8], showing that in principle ideal ORE is achievable. However, current multilinear

[6] In [7], it was called efficiently-orderable encryption.

maps are quite inefficient, and moreover have been subject to numerous attacks (e.g. [16, 32, 17]).

In order to develop efficient schemes, one can relax the security requirements to allow for more leakage. Order-preserving encryption (OPE) [1, 6] — which actually predates ORE — is one example, where Comp is simply integer comparison. Very efficient constructions of OPE are known [6]. However, OPE necessarily leaks much more information about the plaintexts [6] than ideal ORE; intuitively, the difference between ciphertexts can be used to approximate the difference between the plaintexts. More recently, there have been efforts to achieve better security without sacrificing too much efficiency: Chenette, Lewi, Weis and Wu (CLWW) [15] recently gave an ORE construction which leaks only the position of the most significant differing bits of the plaintexts.

Unfortunately, even hypothetical ideal ORE has recently been shown insecure for various use cases [24, 25, 3, 19, 30, 26, 10, 34, 20, 22]. This is even if the scheme itself reveals nothing but the order of the plaintexts. The problem is that just the order of plaintexts alone can already reveal a significant amount of information about the data. For example, if the data is chosen uniformly from the entire domain, then even *ideal* ORE will leak the most significant bits. As the most significant bits are often the most important ones, this is troubling.

The problem is that the definitions of ORE, while precise and provable, do not immediately provide any "semantically meaningful" guarantees for the privacy of the underlying data. Indeed, the above attacks show that when the adversary has a strong estimate of the prior distribution the data is drawn from, essentially no security is possible. However, we contend that there are scenarios (see below) where the adversary lacks this knowledge. A core problem in such scenarios is that the privacy of one message is inherently dependent on what other ciphertexts the adversary sees. Analyzing these correlations under arbitrary sources of data, even for ideal ORE, can be quite difficult. Only very mild results are known, for example the fact that either CLWW leakage or ideal leakage provably hides the *least* significant bits of uniformly chosen data. Unfortunately, these bits are probably of less importance (e.g. for salaries).

Therefore, a central goal of this paper is to devise a semantically meaningful notion of privacy for the underlying data in the case that the adversary does not have a strong estimate of the prior distribution, and develop a construction attaining this notion not based on multilinear maps.

We stress that we are not trying to devise a scheme that is secure in the use cases of the attacks above, as many of the attacks above would apply to *any* ORE scheme; we are instead aiming to identify settings where the attacks do not apply, and then provide a scheme satisfying a given notion of security in this setting.

1.1 This Work: Parameter-Hiding ORE

In this work, we give one possible answer to the question above. Rather than focusing on the individual data records, we instead ask about the privacy of the

distribution they came from. We show how to protect some information about the underlying data distribution.

Motivating Example. To motivate our notion, consider the following setting. A large university wants to outsource its database of student GPAs. For simplicity, we will assume each student's academic ability is independent of other students, and that this is reflected in the GPA. Thus, we will assume that each GPA is sampled independently and identically according to some underlying distribution. The university clearly wants to keep each individual's GPA hidden. It also may want aggregate statistics such as mean and variance to be hidden, perhaps to avoid getting a reputation for handing out very high or very low grades.

Distribution-Hiding ORE. This example motivates a notion of distribution-hiding ORE, where all data is sampled independently and identically from some underlying distribution D, and we wish to hide as much as possible about D. We would ideally like to handle arbitrary distributions D, but in many cases will accept handling certain special classes of distributions. Notice that if the distribution itself is completely hidden, then so too is every individual record, since any information about a record is also information about D.

We begin with the following trivial observation: if D has high min-entropy (namely, super-logarithmic), then the ideal ORE leakage is just a random ordering with no equalities, since there are no collisions with overwhelming probability. In particular, this leakage is independent of the distribution D; as such, ideal ORE leakage hides everything about the underlying distribution, except for the super-logarithmic lower bound on min-entropy. Thus, we can use the multilinear map-based scheme of [8] to achieve distribution-hiding ORE for any distribution with high min-entropy.

We note the min-entropy requirement is critical, since for smaller min-entropies, the leakage allows for determining the frequency of the most common elements, hence learning non-trivial information about D.[7]

Unfortunately, the only way we know to build distribution-hiding ORE is using ideal leakage as above; as such, we do not know of a construction not based on multilinear maps. Instead, in hopes of building such a scheme, we will allow some information about the distribution to leak.

Parameter-Hiding ORE. We recall that in many settings, data follows a known type of distribution. For example, the central limit theorem implies that many quantities such as various physical, biological, and financial quantities are (approximately) normally distributed. It is also common practice to assign grades

[7] This min-entropy requirement may be somewhat problematic in some settings. GPAs for example, probably have fewer than 10 bits of entropy. However, adding small random noise to the data before encrypting (much smaller than the precision of the data) will force the data to have high min-entropy without changing the order of data, with the exception that identical data will appear different when comparing. In many cases (such as answering range queries) it is totally acceptable to fail to identify identical data.

on an approximately normal distribution, so GPAs might reasonably be conjectured to be normal. For a different example, insurance claims are often modeled according to the Gamma distribution.

Therefore, since the general shape of the distribution is typically known, a reasonable relaxation of distribution-hiding ORE is what we will call *parameter-hiding ORE*. Here, we will assume the distribution has a *known, public* "shape" (*e.g.* normal, uniform, Laplace *etc.*) but it may be shifted or scaled. We will allow the overall shape to be revealed; our goal instead is to completely hide the shifting and scaling information. More precisely, we consider a distribution D over $[0, 1]$ which will describe the general shape of the family of distributions in question. For example, if the shape in consideration is the set of uniform distributions over an interval, we may take D to be uniform distribution over $[0, 1]$; if the shape is the normal distribution, we will take D be be the normal distribution with mean $1/2$, and standard deviation small enough so that the vast majority of the mass is in $[0, 1]$. Let $D_{\alpha,\beta}$ be the distribution defined as: first sample $x \leftarrow D$, and then output $\lfloor \alpha x + \beta \rfloor$. We will call α the scaling term and β the shift. The adversary receives a polynomial number of encryptions of plaintext sampled iid from $D_{\alpha,\beta}$ for some α, β. We will call a scheme *parameter hiding* if the scale and shift are hidden from any computationally bounded adversary. Our main theorem is that it is possible to construct such parameter-hiding ORE from bilinear maps:

Theorem 1 (Informal). *Assuming bilinear maps, it is possible to construct parameter-hiding ORE for any "smooth" distribution D, provided the scaling term is "large enough."*

We note the restrictions to large scalings are inherent: any small scaling will lead to a distribution with low min-entropy. As discussed above, even with ideal ORE, it is possible to estimate the min-entropy of low min-entropy distributions, and hence it would be possible to recover the scaling term if the scaling term is small. Some restrictions on the shape of D are also necessary, as certain shapes can yield low min-entropy even for large scalings. "Smoothness" (which we will define as having a bounded derivative) guarantees high min-entropy at large scales, and is also important technically for our analysis.

1.2 Technical Overview

As a starting point, we will consider the leakage profile of Chenette, Lewi, Weis and Wu [15] (henceforth referred to as CLWW), which reveals the position of the most significant differing bit between any two plaintexts. This is quite a lot of information: for example, it can be used to get rough bounds on the difference between two plaintexts. Thus, CLWW cannot be parameter hiding, since the scaling term is not hidden. However, CLWW will be a useful starting point, as it will allow us to construct *shift-hiding* ORE, where we only care about hiding the shift term. To help illustrate our approach, we will therefore first describe an equivalent formulation of CLWW leakage, which we will then explain how to extend to get full parameter-hiding ORE.

An alternative view of CLWW leakage Consider the plaintext space $\{0, 1, 2, \ldots, 2^\ell - 1\}$. We will think of the plaintexts as leaves in a full binary tree of depth ℓ. In this tree, the position of the most significant differing bit between two plaintexts corresponds to the depth of their nearest ancestor. The leakage of CLWW can therefore can be seen as revealing the tree consisting of all given plaintexts, their ancestors in the tree up to the lowest common ancestor, and the order of the leaves, with all other information removed. See Figure 1 for an illustration.

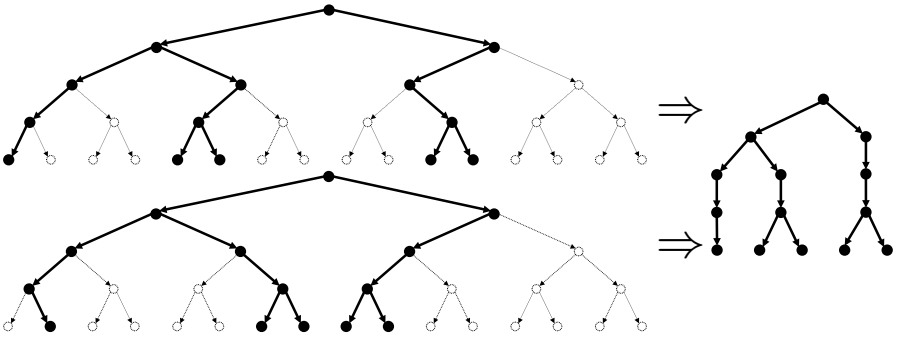

Fig. 1: CLWW Leakage. The two sets of plaintext $\{0, 4, 5, 10, 11\}$ and $\{1, 6, 7, 8, 9\}$ correspond to equivalent subtrees. If the message space extends beyond 15, the CLWW leakage remains the same as depicted, since the leakage only reveals the tree up to the most recent ancestor.

Now, suppose all plaintext elements are in the range $[0, 2^i)$ for some i. This means they all belong in the same subtree at height i; in particular, the CLWW leakage will only have depth at most i. Now, suppose we add a multiple of 2^i to every plaintext. This will simply shift all the plaintexts to being in a different subtree, but otherwise keep the same structure. Therefore, the CLWW leakage will remain the same.

Therefore, while CLWW is not shift hiding, it is *shift periodic*. In particular, if imagine a distribution D whose support is on $[0, 2^i)$, and consider shifting D by β. Consider an adversary A, which is given the CLWW leakage from q plaintexts sampled from the shifted D, and outputs a bit. If we plot the probability $p(\beta)$ that A outputs 1 as a function of β, we will see that the function is periodic with period 2^i.

Shift-Hiding ORE/OPE. With this periodicity, it is simple to construct a scheme that is shift hiding. To get a shift-hiding scheme for message space $[0, 2^\ell)$, we instantiate CLWW with message space $[0, 2^{\ell+1})$. We also include as part of the secret key a random shift γ chosen uniformly in $[0, 2^\ell)$. We then encrypt a message m as $\mathsf{Enc}(m + \gamma)$. Adding a random shift can be seen as convolving the signal $p(\beta)$ with the rectangular function

$$q(\beta) = \begin{cases} 2^{-\ell} & \text{if } \beta \in [0, 2^\ell) \\ 0 & \text{otherwise} \end{cases}$$

Since the rectangular function's support matches the period of p, the result is that the convolved signal $\hat{p}$ is *constant*. In other words, the adversary always has the same output distribution, regardless of the shift β. Thus, we achieve shift hiding.

When the comparison algorithm of an ORE scheme is simple integer comparison, we say the scheme is an *order-preserving encryption* (OPE) scheme. OPE is preferable because it can be used with fewer modifications to a database server. We recall that CLWW can be made into an OPE scheme — where ciphertexts are integers and comparision is integer comparison — while maintaining the CLWW leakage profile. Our conversion to shift-hiding preserves the OPE property, so we similarly achieve a shift-hiding OPE scheme.

Scale-Hiding ORE/OPE. We note that we can also turn any shift-hiding ORE into a scale-hiding ORE. Simply take the logarithm of the input before encrypting; now multiplying by a constant corresponds to shifting by a constant. Of course, taking the logarithm will result in non-integers; this can easily be fixed by rounding to the appropriate level of precision (enough precision to guarantee no collisions over the domain) and scaling up to make the plaintexts integral. Similarly, we can also obtain scale-hiding OPE if we start with an OPE scheme.

Impossibility of parameter-hiding OPE. One may hope to achieve both shift-hiding and scale-hiding by some combination of the two above schemes. For example, since order *preserving* encryption schemes can be composed, one can imagine composing a shift-hiding scheme with a scale-hiding scheme. Interestingly, this does not give a parameter-hiding scheme. The reason is that shifts/scalings of the plaintext do not correspond to shifts/scalings of the ciphertexts. Therefore, while the outer OPE may provide, say, shift-hiding for its inputs, this will not translate to shift-hiding of the inner OPE's inputs.

Nonetheless, one may hope that tweaks to the above may give a scheme that is simultaneously scale and shift hiding. Perhaps surprisingly, we show that this is actually impossible. Namely, we show that *OPE cannot possibly be parameter-hiding*. Due to space limit, we put the rigorous proof in our full version [12].

This impossibility shows that strategies leveraging CLWW leakage are unlikely to yield parameter-hiding ORE schemes. Interestingly, all ORE schemes we are aware of that can be constructed from symmetric crypto can also be made into OPE schemes. Thus, this suggests we need stronger tools than those used by previous efficient schemes.

Parameter Hiding via Smoothed CLWW Leakage Motivated by the above, we must seek a different leakage profile if we are to have any hope of achieving parameter-hiding ORE. We therefore first describe a "dream" leakage that will allow us to perform similar tricks as in the shift hiding case in order to achieve both scale and shift hiding simultaneously. Our dream leakage will be a "smoothed" CLWW leakage, where all nodes of degree exactly 2 are replaced with an edge between the two neighbors. In other words, the dream leakage is

the smallest graph that is "homeomorphic" to the CLWW leakage. See Figure 2 for an illustration.

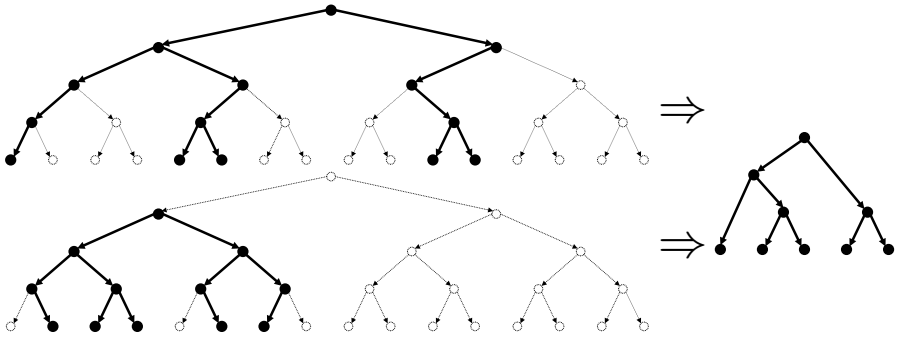

Fig. 2: Smoothed CLWW Leakage. The two sets of plaintext $\{0, 4, 5, 10, 11\}$ and $\{1, 2, 3, 5, 6\}$ correspond to equivalent smoothed subtrees. Notice that the CLWW leakage for these two trees is different.

Our key observation is that this smoothed CLWW leakage now exhibits additional periodicity. Namely, if we multiply every plaintext by 2, every edge in the bottom layer of the CLWW leakage will get subdivided into a path of length 2, but smoothing out the leakage will result in the same exact graph. This means that smoothed CLWW leakage is periodic in the *log* domain.

In particular, consider a distribution D with support on $[0, 2^i)$, and suppose it is multiplied by α. Consider an adversary A, which is given the smoothed CLWW leakage from q plaintexts sampled from a scaled D, and outputs a bit. If we plot the probability $p(\log_2 \alpha)$ that A outputs 1 as a function of α, we will see that the function is periodic with period 1.

Therefore, we can perform a similar trick as above. Namely, we convolve p with the uniform distribution over the period of p in the log domain. We accomplish this by including a random scalar α as part of the secret key, and multiplying by α before encrypting. However, this time several things are different:

- Since we are working in the log domain, the logarithm of the random scalar α has to be uniform. In other words, α is log-uniform
- Since we are working over integers instead of real numbers, many issues arise.
  - First, α needs to be an integer to guarantee that the scaled plaintexts are still integers. This means we cannot choose α at log-uniformly over a single log period, since then α only has support on $\{1, 2\}$. Instead, we need to choose α log-uniformly over a sufficiently large multiple of the period that α approximates the continuous log-uniform distribution sufficiently well.
  - Second, unlike the shift case, sampling at random from D and then scaling is not the same as sampling from a scaled version of D, since the

rounding step does not commute with scaling. For example, for concreteness consider the normal distribution. If we sample from a normal distribution (and round) and then scale, the resulting plaintexts will all be multiples of α. However, if we sample directly from a scaled normal distribution (and then round), the support of the distribution will include integers which are not multiples of α.

To remedy this issue, we observe that if the plaintexts are sampled from a wide enough distribution, their differing bits will not be amongst the lowest significant bits. Hence, the leakage will actually be independent of the lower order bits. For example, this means that while the rounding does not commute with the scaling, the leakage actually does not depend on the order in which the two operations are carried out.

- The above arguments can be made to work for, say, the normal distribution. However, we would like to have a proof that works for any distribution. Unfortunately, for distributions that oscillate rapidly, we may run into trouble with the above arguments, since rounding such distributions can cause odd behaviors at all scales. This problem is actually unavoidable, as quickly oscillating distributions may have actually have low min-entropy even at large scales. Therefore, we must restrict to "smooth" functions that have a bounded derivative.

Using a careful analysis, we are able to show for smooth distributions that we achieve the desired scale hiding.

– Finally, we want to have a scheme that is both scale and shift hiding. This is slightly non-trivial, since once we introduce, say, a random shift, we have modified the leakage of the scheme, and cannot directly appeal to the arguments above to obtain scale hiding as well. Instead, we distill a set of specific requirements on the leakage that will work for both shift hiding and scale hiding. We show that our shift hiding scheme above satisfies the requirements needed in order for us to introduce a random scale and additionally prove scale hiding.

Achieving Smoothed CLWW Leakage. Next we turn to actually constructing ORE with smoothed CLWW leakage. Of course, ideal ORE has better than (smoothed) CLWW leakage, so we can construct such ORE based on multilinear maps. However, we want a construction that uses standard tools.

We therefore provide a new construction of ORE using pairings that achieves smoothed CLWW leakage. We believe this construction is of interest on its own, as it is achieves the to-date smallest leakage of any non-multilinear-map-based scheme.

CLWW ORE and how to reduce its leakage. Our construction builds on the ideas of CLWW, so we first briefly recall the ORE scheme of CLWW. In their (basic) scheme, the encryption key is just a PRF key K. To encrypt a plaintext $x \in \{0,1\}^n$, for each prefix $p_i = x[1,\ldots,i]$, the scheme computes

$$y_i = \mathsf{PRF}_K(p_i) + x_{i+1}$$

where x_{i+1} is the $(i+1)$-st bit of x, and the output of $\mathsf{PRF} \in \{0,1\}^\lambda$ is treated as an integer (we will take λ to be the security parameter). The ORE ciphertext is then $(y_1 \ldots, y_n)$. To compare two ciphertexts $(y_1 \ldots, y_n)$ and $(y'_1 \ldots, y'_n)$, one finds the smallest index i such that $y_i \neq y'_i$, and outputs 1 if $y'_i - y_i = 1$. This naturally reveals the index of the bit where the plaintexts differ.

Our approach to reducing the leakage is to attempt to hide the index i where the plaintexts differ. As a naive attempt at this, first consider what happens if we modify the scheme to simply randomly permute the outputs $(y_1 \ldots, y_n)$ (with a fresh permutation chosen for each encryption). We can still compare ciphertexts by appropriately modifying the comparison algorithm: now given $c = (y_1 \ldots, y_n)$ and $c' = (y'_1 \ldots, y'_n)$ (permuted as above), it will look for indices i, j such that either $y'_i - y_j = 1$, in which case it outputs 1, or $y_j - y'_i = 1$, in which case it outputs 0. (If we choose the output length of the PRF to be long enough then this check will be correct with overwhelming probability.)

This modification, however, does not actually reduce leakage: an adversary can still determine the most significant differing bit by counting how many elements c and c' have in common.

We can however recover this approach by preventing an adversary from detecting how many elements c and c' have in common. To do so, we introduce and employ the new notion of *property-preserving hashing* (PPH). Intuitively, a PPH is a randomized hashing scheme that is designed to publicly reveal a particular predicate P on pairs of inputs.

PPH can be seen as the hashing (meaning, no decryption) analogue of the notion of property-preserving encryption, a generalization of order-revealing encryption to arbitrary properties due to Pandey and Rouselakis [35]. (This can also be seen as a symmetric-key version of the notion of "relational hash" due to Mandal and Roy [31].)

Specifically, we construct and employ a PPH for the property

$$P_1(x, x') = \begin{cases} 1 & \text{if } x = x' + 1 \\ 0 & \text{otherwise} \end{cases}$$

(Here x, x' are not plaintexts of the ORE scheme, think of them as other inputs determined below.) Security requires that this is *all* that is leaked; in particular, input equality is *not* leaked by the hash values (which requires a randomized hashing algorithm).

Now, the idea is to modify the scheme to include a key K_H for such a PPH $\mathcal{H}$, and the encryption algorithm to not only randomly permute the y_i's but hash them as well, i.e., output $(h_1, \ldots, h_n)$ where $h_i = \mathcal{H}_{K_H}(y_i)$ for the permuted y_i's.[8] The comparison algorithm can again be modified appropriately, namely to not to check if $y'_i - y_j = 1$ but rather if their h'_i and h'_j hash values satisfy P_1 via the PPH (and similarly for the check $y_j - y'_i = 1$).

[8] A minor issue here is that we now lose decryptability for the resulting ORE scheme; however, this can easily be added back in a generic way by also encrypting the plaintext separately under a semantically secure scheme.

For any two messages, the resulting ORE scheme is actually ideal: it only reveals the order of the underlying plaintexts, but nothing else. However, for three messages m, m', m'' we see that some additional information is leaked. Namely, if we find that $y'_i - y_j = 1$ $y''_k - y_j = 1$, then we know that $y'_j = y''_k$. We choose the range of the PRF large enough so that this can only happen if y'_j and y'_k are both $\mathsf{PRF}_K(p_\ell) + x_{\ell+1}$ for the same prefix p_ℓ and same bit $x_{\ell+1}$, and y'_j corresponds to the most significant bit where m' differs from m, y''_k corresponds to the most significant bit where m'' differs from m, and moreover these positions are the same. Therefore, the adversary learns whether these most-significant differing bits are the same. It is straightforward to show that this leakage is exactly equivalent to the smoothed CLWW leakage we need. Proving this ORE scheme secure wrt. this leakage based on an achievable notion of security for the PPH turns out to be technically challenging. Nevertheless, we manage to prove it "non-adaptively secure," meaning the adversary is required to non-adaptively choose the dataset, which is realistic for a passive adversary in the outsourced database setting.

Property-preserving hash from bilinear maps. Next we turn to constructing a property-preserving hash (PPH) for the property $P_1(x, x') = x = x' + 1$. For this, we adapt techniques from perfectly one-way hash functions [9, 31] to the symmetric-key setting and use asymmetric bilinear groups. Roughly, in our construction the key for the hash function is a key K for a pseudorandom function PRF and, letting $e\colon G_1 \times G_2 \to G_T$ be an asymmetric bilinear map on prime order cyclic groups G_1, G_2 with generators g_1, g_2, the hash of x is

$$\mathcal{H}_K(x) = (g_1^{r_1}, g_1^{r_1 \mathsf{PRF}_K(x)}, g_2^{r_2}, g_2^{r_2 \mathsf{PRF}_K(x+1)})$$

for fresh random $r_1, r_2 \in \mathbb{Z}_p$. (Thus, the PRF is also pushed to our PPH construction and can be dropped from from the higher-level ORE scheme when our hash function is plugged-in.) The bilinear map allows testing whether $P_1(x, x')$ from $\mathcal{H}_K(x), \mathcal{H}_K(x')$, and intuitively our use of asymmetric bilinear groups prevents testing other relations such as equality (formally we use the XSDH assumption). We prove the construction secure under an indistinguishability-based notion in which the adversary has to distinguish between the hash of a random challenge x^* and a random hash value, and can query for hash values of inputs x of its choice as long as $P_1(x, x^*)$ and $P_1(x^*, x)$ are both 0. Despite being restricted,[9], this notion suffices in our ORE scheme above.

When our PPH is plugged into our ORE scheme, ciphertexts consist of $4n$ group elements, and order comparison requires $n(n-1)$ pairing computations on average. We also note that CLWW gave an improved version of their scheme where ciphertexts are size $O(n)$ rather than $O(n\lambda)$ for security parameter λ, however, we have reason to believe this may be difficult for schemes with our improved leakage profile, see below.

[9] More generally, following [35] one could allow the adversary to choose two challenge inputs and make queries that do not allow it to trivially distinguish them, but we are unable to prove our construction secure under this stronger notion.

Piecing everything together, we obtain a parameter-hiding ORE from bilinear maps. We note that, as parameter-hiding OPE is impossible, we achieve the first construction of ORE without multilinear maps secure with a security notion that is impossible for OPE.

Generalizing our ORE scheme. In our full version [12], we also show several extensions to our smoothed CLWW ORE scheme. In one direction, we achieve an improved level of leakage by considering blocks of bits at a time(encrypting message block by block, rather than bit by bit). We show that if the block size is only 2, then we improve security and efficiency simultaneously, while for larger block sizes the leakage continues to reduce but the efficiency compared to the basic scheme (in terms of both ciphertext size and pairings required for comparison) decreases.

On the other direction, we also show how to improve efficiency while sacrificing some security. We give a more efficient version of the scheme than above (only need $O(n)$ pairings for each comparison), that is still sufficient for achieving parameter-hiding ORE using our conversion.

In addition, we also show how our ORE scheme easily gives a *left/right ORE* as defined by [29] that also improves on their leakage. In left/right ORE, ciphertexts can be generated in either the left mode or right mode, and the comparison algorithm only compares a left and a right ciphertext. Security requires that no information is leaked amongst left and right ciphertexts in isolation.

1.3 Discussion and Perspective

The original OPE scheme of [6] leaks "whatever a random order-preserving function leaks." Unfortunately, this notion does not say anything about what such leakage actually looks like. The situation has been improved in recent works on OPE such as CLWW which define a precise "leakage profile" for their scheme. However, such leakage profiles are still of limited use, since they do not obviously say anything about the actual privacy of the underlying data.

We instead study ORE with a well-defined privacy notion for the underlying plaintexts. A key part of our results is showing how to translate sufficiently strong leakage profiles into such privacy notions. Nonetheless, we do not claim that our new ORE scheme is safe to use in general higher-level protocols. We only claim security as long all that is sensitive is the scale and shift of the underlying plaintext distributions. If, for example, if the shape of the distribution is highly sensitive, or if there are correlations to other data available to the attacker, our notion is insufficient.

However, our construction provably has better leakage than existing efficient schemes, and it at least shows some meaningful security for specific situations. Moreover we suspect that the scheme can be shown to be useful in many other settings by extending our techniques.

1.4 Related Work

Work done on "leaky cryptography" includes work on multiparty computation [33], searchable symmetric and structured encryption [37, 21, 13, 18, 14, 11, 28], and property-preserving encryption [5, 6, 35]. In the database community, the problem of querying an encrypted database was introduced by Hacigümüş, Iyer, Li and Mehrotra [23], leading to a variety of proposals there but mostly lacking formal security analysis. Proposals of specific outsourced database systems based on property-preserving encryption like ORE include CryptDB [36], Cipherbase [2], and TrustedDB [4].

Besides, in [29], the authors give an efficient ORE construction based on PRFs, while their leakage profile cannot achieve shift hiding and scale hiding simultaneously, which means their scheme cannot meet our privacy notion. Moreover, in [27], the authors give an alternative ORE construction, based on on function revealing encryption for simple functions, namely orthogonality testing and intersection cardinality, while their leakage needs further analysis.

2 Background

NOTATION. All algorithms are assumed to be polynomial-time in the security parameter (though we will sometimes refer to efficient algorithms explicitly). We will denote the security parameter by λ. For a random variable Y, we write $y \xleftarrow{\$} Y$ to denote that y is sampled according to Y's distribution, moreover, let D be Y's distribution, we abuse notation $y \xleftarrow{\$} D$ to mean that y is sampled according to D. For an algorithm A, by $y \xleftarrow{\$} A(x)$ we mean that A is executed on input x and the output is assigned to y, furthermore, if A is randomized, then we write $y \xleftarrow{\$} \mathcal{A}(x)$ to denote running $\mathcal{A}$ on input x with a fresh random tape and letting y be the random variable induced by its output. We denote by $\Pr[A(x) = y : x \xleftarrow{\$} X]$ the probability that A outputs y on input x when x is sampled according to X. We say that an adversary $\mathcal{A}$ has advantage ϵ in distinguishing X from Y if $\Pr[A(x) = 1 : x \xleftarrow{\$} X]$ and $\Pr[A(y) = 1 : y \xleftarrow{\$} Y]$ differ by at most ϵ.

When more convenient, we use the following probability-theoretic notation instead. We write $P_X(x)$ to denote the probability that X places on x, i.e. $P_X(x) = \Pr[X = x]$, and we say $P_X(x)$ is the probability density function (PDF) of X's distribution. The *statistical distance* between X and Y is given by $\Delta = \frac{1}{2}\sum_x |P_X(x) - P_Y(x)|$. If $\Delta(X, Y)$ is at most ϵ then we say X, Y are ϵ-close. It is well-known that if X, Y are ϵ-close then any (even computationally unbounded) adversary A has advantage at most ϵ in distinguishing X from Y.

The *min-entropy* of a random variable X is $H_\infty(X) = -\log(\max_x P_X(x))$. A value $\nu \in \mathbb{R}$ depending on λ is called *negligible* if its absolute value goes to 0 faster than any polynomial in λ, i.e. $\forall c > 0\ \exists \lambda^* \in \mathbb{N}\ \forall \lambda \geq \lambda^* : |\nu| \leq \frac{1}{\lambda^c}$. We let $[M] = \{1, \ldots, M\}$, $[M]' = \{0, \ldots, M-1\}$ and $[M, N] = \{M, \ldots, N\}$. We write $\boldsymbol{m}$ as a vector of plaintexts and $|\boldsymbol{m}|$ as the vector's length, namely

$\boldsymbol{m} = (m_1, \ldots, m_s)$ and $|\boldsymbol{m}| = s$. For a vector $\boldsymbol{m}$, by $a\boldsymbol{m}$ we mean $(am_1, \ldots, am_s)$ and we write $\boldsymbol{m}+b$ to denote $(m_1+b, \ldots, m_s+b)$. Let x be a real number, we write $\lfloor x \rfloor$ as the largest integer s.t. $\lfloor x \rfloor \leq x$, and $\lceil x \rceil$ as the smallest integer s.t. $\lceil x \rceil \geq x$. By $\lfloor x \rceil$, we mean rounding x to the nearest integer, namely $-1/2 \leq \lfloor x \rceil - x < 1/2$. If P is a predicate, we write $\mathbf{1}(P)$ for the function that takes the inputs to P and returns 1 if P holds and 0 otherwise.

PRFs. We use the standard notion of a PRF. A function $F : \{0,1\}^\lambda \times D \to \{0,1\}^\lambda$ is said to be a *PRF with domain D* if for all efficient $\mathcal{A}$ we have that

$$|\Pr[\mathcal{A}^{F(K,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{g(\cdot)}(1^\lambda) = 1]|$$

is a negligible function of λ, where K is uniform over $\{0,1\}^\lambda$ and g is uniform over all functions from D to $\{0,1\}^\lambda$.

ORE. The following definition of syntax for order-revealing encryption makes explicit that comparison may use helper information (e.g. a description of a particular group) by incorporating a *comparison key*, denote ck.

Definition 2 (ORE). *A* ORE scheme *is a tuple of algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ with the following syntax.*

- *The key generation algorithm $\mathcal{K}$ is randomized, takes inputs $(1^\lambda, M)$, and always emits two outputs $(\mathsf{sk}, \mathsf{ck})$. We refer to the first output sk as the* secret key *and the second output ck as the* comparison key*.*
- *The encryption algorithm $\mathcal{E}$ is randomized, takes inputs (sk, m) where $m \in [M]$, and always emits a single output c, that we refer to as a* ciphertext*.*
- *The comparison algorithm $\mathcal{C}$ is deterministic, takes inputs (ck, c_1, c_2), and always emits a bit.*

If the comparison algorithm $\mathcal{C}$ is simple integer comparison (i.e., if $\mathcal{C}(\mathsf{ck}, c_1, c_2)$ is a canonical algorithm that treats its the ciphertexts and binary representations of integers and tests which is greater) then the scheme is said to be an order-preserving encryption (OPE) *scheme.*

Correctness of ORE schemes. Intuitively, an ORE scheme is correct if the comparison algorithm can output the order of the underlying plaintext, by taking ck and two ciphertexts as inputs.

Our constructions will only be *computationally* correct, i.e. correct with overwhelming probability when the input messages are provided by an efficient process, under hardness assumptions. Formally, we define correctness using the game $\mathrm{COR}^{\mathsf{ore}}_{\Pi}(\mathcal{A})$, which is defined as follows: The game starts by running $(\mathsf{sk}, \mathsf{ck}) \xleftarrow{\$} \mathcal{K}(1^\lambda, M)$, and it gives ck to $\mathcal{A}$. The adversary $\mathcal{A}$ then outputs two messages $x, y \in [M]$. The game computes $c_1 \xleftarrow{\$} \mathcal{E}(\mathsf{sk}, x)$ and $c_2 \xleftarrow{\$} \mathcal{E}(\mathsf{sk}, y)$, outputs 1 if $x < y$ but $\mathcal{C}(\mathsf{ck}, c_1, c_2) = 0$.

We say that an ORE scheme Π is *computationally correct* if for all efficient adversaries $\mathcal{A}$, all $M = \mathrm{poly}(\lambda)$, we have that $\Pr[\mathrm{COR}^{\mathsf{ore}}_{Pi}(\mathcal{A}) = 1]$ is a negligible function in the security parameter.

SECURITY OF ORE SCHEMES. The following simulation-based security definition is due to Chenette et al. [15]. Here a *leakage profile* is any randomized algorithm. The definition refers to games given in Figure 3, which we review now. In the real game, key generation is run and the adversary is given the comparison key and oracle access to the encryption algorithm with the corresponding secret key. The adversary eventually outputs a bit that the game uses as its own output. In the ideal simulation game, the adversary is interacting with the same oracle, but the comparison key is generated by a stateful simulator, and the oracle responses are generated by the simulator which receives leakage from the stateful leakage algorithm $\mathcal{L}$.

Game $\mathrm{REAL}^{\mathsf{ore}}_{\Pi}(\mathcal{A})$:	Game $\mathrm{SIM}^{\mathsf{ore}}_{\Pi,\mathcal{L}}(\mathcal{A},\mathcal{S})$:
$(\mathsf{sk},\mathsf{ck}) \xleftarrow{\$} \mathcal{K}(1^\lambda, M)$; $b \xleftarrow{\$} \mathcal{A}^{\textsc{Enc}}(\mathsf{ck})$	$\mathsf{st}_\ell \leftarrow \bot$; $(\mathsf{ck},\mathsf{st}_s) \xleftarrow{\$} S(1^\lambda)$; $b \xleftarrow{\$} \mathcal{A}^{\textsc{Enc}}(\mathsf{ck})$
Return b	Return b
$\textsc{Enc}(m)$:	$\textsc{Enc}(m)$:
Return $\mathcal{E}(\mathsf{sk}, m)$	$(L,\mathsf{st}_\ell) \xleftarrow{\$} \mathcal{L}(\mathsf{st}_\ell, m)$; $(c,\mathsf{st}_s) \xleftarrow{\$} \mathcal{S}(L,\mathsf{st}_s)$
	Return c

Fig. 3: Games $\mathrm{REAL}^{\mathsf{ore}}\Pi(\mathcal{A})$ (left) and $\mathrm{SIM}^{\mathsf{ore}}_{\Pi,\mathcal{L}}(\mathcal{A},\mathcal{S})$ (right), where $\Pi = (\mathcal{E},\mathcal{C})$ is an ORE scheme, $\mathcal{L}$ is a leakage profile, $\mathcal{A}$ is an adversary, and $\mathcal{S}$ is a simulator.

Definition 3 ($\mathcal{L}$-simulation-security for ORE). *For an ORE scheme Π, an adversary $\mathcal{A}$, a simulator $\mathcal{S}$, and leakage profile $\mathcal{L}$, we define the games* $\mathrm{REAL}^{\mathsf{ore}}_{\Pi}(\mathcal{A})$ *and* $\mathrm{SIM}^{\mathsf{ore}}_{\Pi,\mathcal{L}}(\mathcal{A})$ *in Figure 3. The advantage of $\mathcal{A}$ with respect to $\mathcal{S}$ is defined as*

$$\mathsf{Adv}^{\mathsf{ore}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = \left|\Pr[\mathrm{REAL}^{\mathsf{ore}}_{\Pi}(\mathcal{A}) = 1] - \Pr[\mathrm{SIM}^{\mathsf{ore}}_{\Pi,\mathcal{L}}(\mathcal{A},\mathcal{S}) = 1]\right|.$$

We say that Π is $\mathcal{L}$-simulation-secure if for every efficient adversary $\mathcal{A}$ there exists an efficient simulator $\mathcal{S}$ such that $\mathsf{Adv}^{\mathsf{ore}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda)$ is a negligible function.

We also define non-adaptive *variants of the games where $\mathcal{A}$ gets a single query to an oracle that accepts a vector of messages of unbounded size. In the real game* $\mathrm{REAL}^{\mathsf{ore\text{-}na}}_{\Pi}(\mathcal{A})$*, the oracle returns the encryptions applied independently to each message. In the ideal game* $\mathrm{SIM}^{\mathsf{ore\text{-}na}}_{\Pi}(\mathcal{A})$*, the leakage function gets the entire vector of messages as input and produces an output L that is then given to $\mathcal{S}$ which produces a vector of ciphertexts, which are returned by the oracle.*

We define the non-adaptive advantage *of $\mathcal{A}$ with respect to $\mathcal{S}$ analogously, and denote it* $\mathsf{Adv}^{\mathsf{ore\text{-}na}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda)$. Non-adaptive $\mathcal{L}$-simulation security *is defined analogously.*

Ideal ORE. Ideal ORE is the case where the leakage profile $\mathcal{L}$ is simply the list of results of comparisons between the plaintexts. We note that such a $\mathcal{L}$ is *always* revealed by the comparison algorithm, so ideal ORE is the best one can hope for. Ideal ORE can be constructed from multilinear maps [8].

CLWW Leakage. As an example of a non-ideal leakage profile, consider the leakage $\mathcal{L}_{\mathsf{clww}}$ of Chenette, Lewi, Weis and Wu [15]. For $m_0, m_1 \in \{0,1\}^n$, we define the most significant differing bit of m_1 and m_2, denoted $\mathsf{msdb}(m_0, m_1)$, as the index of first bit where m_0, m_1 differ, or $n+1$ if $m_1 = m_2$.

The CLWW leakage profile $\mathcal{L}_{\mathsf{clww}}$ takes in input a vector of plaintext $\boldsymbol{m} = (m_1, \ldots, m_q)$ and produce the following:

$$\mathcal{L}_{\mathsf{clww}}(m_1, \ldots, m_q) := (\forall 1 \leq i, j \leq n, \mathbf{1}(m_i < m_j), \mathsf{msdb}(m_i, m_j))$$

3 New Security Notions for ORE

In this section, we propose four meaningful notions of privacy: *distribution-hiding*, *parameter-hiding*, *scale-hiding* and *shift-hiding*; in those notions, we are considering the privacy of the underlying *distribution* of data records, rather than the individual data records, and show how to protect information about the underlying data distribution.

Distribution-Hiding for ORE. We assume that all database entries are independently and identically distributed according to some distribution D[10], and the notion of distribution-hiding refers to game defined in Figure 4. In the interactive game, after receiving the public parameter and comparison key, adversary $\mathcal{A}$ picks two distributions D_0, D_1 and sends to challenger $\mathcal{C}$, $\mathcal{C}$ then flips a coin b, samples a sequence of entries from D_b, and sends back the encrypted entries. Eventually $\mathcal{A}$ outputs a bit, and we say adversary wins if it guesses b correctly. We note that if either of D_b has low min-entropy, it is possible for an adversary to estimate the min-entropy by looking for collisions in its ciphertexts. Therefore, we must restrict D_b to have high min-entropy.

Game $\mathsf{DH}_{\Pi,q}(\mathcal{A}, \lambda)$:

$(\mathsf{sk}, \mathsf{ck}) \xleftarrow{\$} \mathcal{K}(1^\lambda, M)$; $D_0, D_1 \leftarrow \mathcal{A}(1^\lambda, \mathsf{ck})$ s.t. $H_\infty(D_b) \geq \omega(\log \lambda)$

$b \xleftarrow{\$} \{0,1\}, \boldsymbol{m} \xleftarrow{\$} D_b$ *s.t.* $|m| = q; \max D_b \leq M; \boldsymbol{c} = \mathcal{E}(\mathsf{sk}, \boldsymbol{m})$

$b' = \mathcal{A}(\mathsf{ck}, \boldsymbol{c})$; Return $(b \stackrel{?}{=} b')$

Fig. 4: Games $\mathsf{DH}_{\Pi,q}(\mathcal{A}, \lambda)$, where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ is an ORE scheme, $q = \mathsf{poly}(\lambda)$, and $\mathcal{A}$ is an adversary.

Definition 4 (Distribution-Hiding for ORE). *For an ORE scheme Π, an adversary $\mathcal{A}$, function $q = q(\lambda)$ we define the games $\mathsf{DH}_{\Pi,q}(\mathcal{A}, \lambda)$ in Figure 4. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{\mathsf{DH}}_{\Pi,q}(\mathcal{A}, \lambda) = |\Pr[\mathsf{DH}_{\Pi,q}(\mathcal{A}, \lambda) - \frac{1}{2}]|$. We say that Π is distribution-hiding if for every efficient adversary $\mathcal{A}$, and any polynomial $q = \mathsf{poly}(\lambda)$, $\mathbf{Adv}^{\mathsf{DH}}_{\Pi,q}(\mathcal{A}, \lambda)$ is a negligible function.*

[10] By D, here we mean a sampling algorithm, such that the outputs of this algorithm obey the distribution D, for ease we denote $\max D$ as the maximum item in D's support.

We immediately observe that ideal ORE achieves distribution hiding, while for other known leakier ORE schemes, it's seems unfeasible to achieve this privacy guarantee. However, in many settings, the general shape of the distribution is often known (that is, if the distribution is normal, uniform, Laplace, etc), and it is reasonable to allow the overall shape to be reveal but hide its mean and/or variance completely, subject to certain restrictions. Before formalize these notion, we firstly introduce some notations.

For a continuous random variable X, where D is X's distribution, we abuse notation $p_D(x) = p_X(x)$. Now we introduce three alternative distributions: $D^{\delta}_{\mathsf{scale}}, D^{\ell}_{\mathsf{shift}}, D^{\delta,\ell}_{\mathsf{aff}}$ with parameter δ, ℓ, where the corresponding probability density function is defined as:

$$p_{D_{\mathsf{scale}}} = \frac{p_D(\frac{x}{\delta})}{\delta};\ p_{D_{\mathsf{shift}}}(x) = p_D(x-\ell);\ p_{D_{\mathsf{aff}}} = \frac{p_D(\frac{x-\ell}{\delta})}{\delta}$$

In other words, $D^{\delta}_{\mathsf{scale}}$ scales the shape of D by a factor of δ; D_{shift} shifts D by ℓ and D_{aff} does both.

Rounded distribution. As our plaintexts are integers, we need map real number to its rounded integer, namely $x \to \lfloor x \rceil$. More precisely, let D be a distribution over real numbers between α and β; we induce a rounded distribution $R^{\alpha,\beta}_D$ on $[\lceil \alpha \rceil, \lfloor \beta \rfloor]$which samples from D and then rounds. Its probability density function is:

$$p_{R^{\alpha,\beta}_D}(k) = \begin{cases} \frac{\int_{\alpha}^{\lceil \alpha \rceil + 1/2} p_D(x)dx}{\int_{\alpha}^{\beta} p_D(x)dx} & k = \alpha \\ \frac{\int_{k-1/2}^{k+1/2} p_D(x)dx}{\int_{\alpha}^{\beta} p_D(x)dx} & k \in [\lceil \alpha + 1 \rceil, \lfloor \beta - 1 \rfloor] \\ \frac{\int_{\lfloor \beta \rfloor - 1/2}^{\beta} p_D(x)dx}{\int_{\alpha}^{\beta} p_D(x)dx} & k = \beta \\ 0 & \textit{Otherwise} \end{cases}$$

In the case of $D^{\delta}_{\mathsf{scale}}$, $D^{\ell}_{\mathsf{shift}}$, or $D^{\delta,\ell}_{\mathsf{aff}}$, we will use the notation $\lfloor D^{\delta}_{\mathsf{scale}} \rceil$, $\lfloor D^{\ell}_{\mathsf{shift}} \rceil$, and $\lfloor D^{\delta,\ell}_{\mathsf{aff}} \rceil$ to denote the respective rounded distributions.

Now, we present the notion "(γ, D)-parameter-hiding" ORE, referring to the game defined in Figure 5. Here, D is a distribution over $[0,1]$, which represents the description of the known shape of the distribution of plaintexts. γ is a lower-bound on the scaling that is allowed. Then key generation is run and adversary is given the public parameter, (γ, D), and the comparison key. Then, the adversary $\mathcal{A}$ sends two pairs of parameters $(\delta_0, \ell_0), (\delta_1, \ell_1)$ to challenger $\mathcal{C}$. Next, $\mathcal{C}$ flips a coin b, checks whether the parameter is proper($\mathbf{1}(\delta_0 \geq \gamma \cap \delta_1 \geq \gamma)$), then samples a sequence of data entries from the rounded distribution $\lfloor D^{\delta_b,\ell_b}_{\mathsf{aff}} \rceil$ and sends back encrypted data. Eventually $\mathcal{A}$ outputs a bit, and we say adversary wins if it guesses b correctly.

Definition 5 ((γ, D)-parameter hiding for ORE). *For an ORE scheme Π, an adversary $\mathcal{A}$, a distribution D, and function $q = q(\lambda)$, we define the games*

Game (γ, D)-$\mathsf{para\text{-}hid}_{\Pi,q}(\mathcal{A}, \lambda)$:
$(\mathsf{sk}, \mathsf{ck}) \xleftarrow{\$} \mathcal{K}(1^\lambda, M)$; $\delta_0, \ell_0, \delta_1, \ell_1 \leftarrow \mathcal{A}(\mathsf{ck}, D)$
If $\delta_0 < \gamma$ or $\delta_1 < \gamma$, output a random bit and abort,
else, $b \xleftarrow{\$} \{0,1\}, \boldsymbol{m} \xleftarrow{\$} \lfloor D_{\mathsf{aff}}^{\delta_b, \ell_b} \rceil, s.t. |m| = q; \max \lfloor D_{\mathsf{aff}}^{\delta_b, \ell_b} \rceil \leq M; \boldsymbol{c} = \mathcal{E}(\mathsf{sk}, \boldsymbol{m})$
$b' = \mathcal{A}(\mathsf{ck}, \boldsymbol{c})$ Return $(b \stackrel{?}{=} b')$

Fig. 5: Games $\mathsf{para\text{-}hid}_{\Pi,q}(\mathcal{A}, \lambda)$, where $\Pi = (\mathcal{E}, \mathcal{C})$ is an ORE scheme, D is a distribution on $[0, 1]$, $\mathcal{A}$ is an adversary

(γ, D)-$\mathsf{para\text{-}hid}_{\Pi,q}(\mathcal{A}, \lambda)$ *in Figure 5. The advantage of* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mathsf{para\text{-}hid}}_{\Pi,q,\gamma,D}(\mathcal{A}, \lambda) = |\Pr[(\gamma, D)\text{-}\mathsf{para\text{-}hid}_{\Pi,q}(\mathcal{A}, \lambda) - \frac{1}{2}]|$$

We say that Π *is* (γ, D)*-parameter hiding if for every efficient adversary* $\mathcal{A}$ *and polynomial* q $\mathbf{Adv}^{\mathsf{para\text{-}hid}}_{\Pi,q,\gamma,D}(\mathcal{A}, \lambda)$ *is a negligible function.*

Similarly, we define (γ, D)-scale hiding and (γ, D)-shift hiding with little change as above. More precisely, in the game of (γ, D)-scale hiding, we add the restriction $\ell_0 = \ell_1 = 0$ and in the game of (γ, D)-shift hiding, we add the restriction $\delta_0 = \delta_1$. Due to the space limit, we skip the formal definitions here.

We note that these three notions are distribution dependent, and we would like they work for any distribution. Unfortunately, quickly oscillating distributions do not fit into our case, as they may have actually low min-entropy for their discretized distributions on integers, even at large scales. Hence, we place additional restrictions. We place the following restriction, which is sufficient, but potentially stronger than necessary:

(η, μ)-SMOOTH DISTRIBUTION. We let D be a distribution where its support mainly on $[0, 1]$ ($\Pr[x \notin [0, 1] : x \leftarrow D] \leq \mathsf{negl}(\lambda)$), we denote $p'_D(x)$ as its derivative, and we say that D is (η, μ)-smooth if 1) $\forall x \in [0, 1], p_D(x) \leq \eta$; 2) $|p'_D(x)| \leq \eta$ for all $x \in [0, 1]$ except for μ points.

Definition 6 ((γ, η, μ)-parameter hiding for ORE). *For an ORE scheme* Π*, we say* Π *is* (γ, η, μ)*-parameter hiding if for every efficient adversary* $\mathcal{A}$*, polynomial* q*, and any* (η, μ)*-smooth distribution* D*,* $\mathbf{Adv}^{\mathsf{para\text{-}hid}}_{\Pi,q,\gamma,D}(\mathcal{A}, \lambda)$ *is a negligible function.*

4 Parameter Hiding ORE

In this section, we will assume we are given an ORE $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ with a "smoothed" version of CLWW leakage, defined below. Later, in Section 5, we will show how to instantiate such a scheme from bilinear maps.

We show how to convert a scheme with smoothed CLWW leakage into a parameter-hiding ORE scheme by simply composing with a linear function: namely, for any plaintext m, the ciphertext has form $\mathcal{E}(\alpha m + \beta)$, where α, β

are the same across all messages and are sampled as part of the secret key. Intuitively, α helps to hide the scale parameter and β hides the shift. We need to be careful about the distributions of α and β; α needs to be drawn from a "discrete log uniform" distribution of appropriate domain, and β needs to be chosen from a uniform distribution of appropriate domain.

The discrete log uniform distribution D on $[A, B]$ ($\mathsf{logU}(A, B)$) has probability density function:

$$p_D(k) = \begin{cases} \frac{1/k}{\sum_{i=A}^{B} 1/i} & i \in [A, B] \\ 0 & Otherwise \end{cases}$$

We say a leakage function $\mathcal{L}$ is smoothed CLWW if:

1. For any two plaintext sequences $\boldsymbol{m}_0, \boldsymbol{m_1}$, if $\mathcal{L}_{\mathsf{clww}}(\boldsymbol{m}_0) = \mathcal{L}_{\mathsf{clww}}(\boldsymbol{m}_1)$, then $\mathcal{L}(\boldsymbol{m}_0) = \mathcal{L}(\boldsymbol{m}_1)$ (in other words, it leaks no more information that CLWW);
2. For any plaintext sequence $\boldsymbol{m}$, $\mathcal{L}(\boldsymbol{m}) = \mathcal{L}(2\boldsymbol{m})$

4.1 Parameter-Hiding ORE

In this part, we give the formal description of parameter-hiding ORE. To simplify our exposition, we first specify some parameters. We will assume we are given:

$$q = \mathsf{poly}(\lambda), M = 2^{\mathsf{poly}(\lambda)}, \gamma = 2^{\omega(\log \lambda)}, \eta, \mu \leq O(1)$$

We will assume γ and M are exactly powers of 2 without loss of generality by rounding up. We define:

$$\tau = \gamma, \xi = \gamma^2, U = 4\xi M, T = \gamma^2 \times U, K = 2 \times T$$

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ be an ORE scheme on message space $[K]$ with smoothed CLWW leakage $\mathcal{L}$. We define our new ORE $\Pi_{\mathsf{aff}} = (\mathcal{K}_{\mathsf{aff}}, \mathcal{E}_{\mathsf{aff}}, \mathcal{C}_{\mathsf{aff}})$ on message space $[M]$ as follows:

- $\mathcal{K}_{\mathsf{aff}}(1^\lambda, M, \Pi)$: On input the security parameter λ, message space $[M]$ and Π, the algorithm picks a super-polynomial $\gamma = 2^{\omega(\log \lambda)}$ as a global parameter, and computes parameters above. Then it runs $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathcal{K}(1^\lambda, K)$, draws $\alpha \xleftarrow{\$} \mathsf{logU}(\xi, 2\xi - 1)$ and β from discrete uniform on $[T]'$ and outputs $\mathsf{sk}_{\mathsf{aff}} = (\mathsf{sk}, \alpha, \beta), \mathsf{ck}_{\mathsf{aff}} = \mathsf{ck}$;
- $\mathcal{E}_{\mathsf{aff}}(\mathsf{sk}_{\mathsf{aff}}, m)$. On input the secret key $\mathsf{sk}_{\mathsf{aff}}$ and a message $m \in [M]$, it outputs

  $$\mathsf{CT}_{\mathsf{aff}} = \mathcal{E}(\alpha m + \beta)$$

  By our choice of message space $[K]$ for Π, the input to $\mathcal{E}$ is guaranteed to be in the message space.
- $\mathcal{C}_{\mathsf{aff}}(\mathsf{ck}_{\mathsf{aff}}, \mathsf{CT}^0_{\mathsf{aff}}, \mathsf{CT}^1_{\mathsf{aff}})$: On inputs the comparison key $\mathsf{ck}_{\mathsf{aff}}$, two ciphertexts $\mathsf{CT}^0_{\mathsf{aff}}, \mathsf{CT}^1_{\mathsf{aff}}$, it outputs $\mathcal{C}(\mathsf{ck}_{\mathsf{aff}}, \mathsf{CT}^0_{\mathsf{aff}}, \mathsf{CT}^1_{\mathsf{aff}})$

Here we also give the description of composted schemes that only achieve "scale-hiding" or "shift-hiding". Formally, we define $\Pi_{\mathsf{scale}} = (\mathcal{K}_{\mathsf{scale}}, \mathcal{E}_{\mathsf{scale}}, \mathcal{C}_{\mathsf{scale}})$ and $\Pi_{\mathsf{shift}} = (\mathcal{K}_{\mathsf{shift}}, \mathcal{E}_{\mathsf{shift}}, \mathcal{C}_{\mathsf{shift}})$, respectively:

- $\mathcal{K}_{\mathsf{scale}}(1^\lambda, M, \Pi)$: On input the security parameter λ, the message space $[M]$ and Π, the algorithm picks a super-polynomial $\gamma = 2^{\omega(\log \lambda)}$ as a global parameter, and computes parameters above. Then it runs $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathcal{K}(1^\lambda, K)$, draws $\alpha \xleftarrow{\$} \mathsf{logU}(\xi, 2\xi - 1)$ and outputs $\mathsf{sk}_{\mathsf{scale}} = (\mathsf{sk}, \alpha), \mathsf{ck}_{\mathsf{scale}} = \mathsf{ck}$;
- $\mathcal{E}_{\mathsf{scale}}(\mathsf{sk}_{\mathsf{scale}}, m)$. On input the secret key $\mathsf{sk}_{\mathsf{scale}}$ and a message $m \in [M]$, it outputs
$$\mathsf{CT}_{\mathsf{scale}} = \mathcal{E}(\alpha m)$$
- $\mathcal{C}_{\mathsf{scale}}(\mathsf{ck}_{\mathsf{scale}}, \mathsf{CT}^0_{\mathsf{scale}}, \mathsf{CT}^1_{\mathsf{scale}})$: On inputs the comparison key $\mathsf{ck}_{\mathsf{scale}}$, two ciphertexts $\mathsf{CT}^0_{\mathsf{scale}}, \mathsf{CT}^1_{\mathsf{scale}}$, it outputs $\mathcal{C}(\mathsf{ck}_{\mathsf{scale}}, \mathsf{CT}^0_{\mathsf{scale}}, \mathsf{CT}^1_{\mathsf{scale}})$.

- $\mathcal{K}_{\mathsf{shift}}(1^\lambda, M, \Pi)$: On input the security parameter λ, the message space $[M]$ and Π, the algorithm picks a super-polynomial $\gamma = 2^{\omega(\log \lambda)}$ as a global parameter, and computes parameters above. Then it runs $(\mathsf{ck}, \mathsf{sk}) \leftarrow \mathcal{K}(1^\lambda)$, draws β from discrete uniform on $[T]'$ and outputs $\mathsf{sk}_{\mathsf{shift}} = (\mathsf{sk}, \alpha), \mathsf{ck}_{\mathsf{shift}} = \mathsf{ck}$;
- $\mathcal{E}_{\mathsf{shift}}(\mathsf{sk}_{\mathsf{shift}}, m)$. On input the secret key $\mathsf{sk}_{\mathsf{shift}}$ and a message $m \in [M]$, it outputs
$$\mathsf{CT}_{\mathsf{shift}} = \mathcal{E}(m + b)$$
- $\mathcal{C}_{\mathsf{shift}}(\mathsf{ck}_{\mathsf{shift}}, \mathsf{CT}^0_{\mathsf{shift}}, \mathsf{CT}^1_{\mathsf{shift}})$: On inputs the comparison key $\mathsf{ck}_{\mathsf{shift}}$, two ciphertexts $\mathsf{CT}^0_{\mathsf{shift}}, \mathsf{CT}^1_{\mathsf{shift}}$, it outputs $\mathcal{C}(\mathsf{ck}_{\mathsf{shift}}, \mathsf{CT}^0_{\mathsf{shift}}, \mathsf{CT}^1_{\mathsf{shift}})$.

The correctness of $\Pi_{\mathsf{aff}}, \Pi_{\mathsf{scale}}$ and Π_{shift} is directly held by correctness of Π, and what is more interesting is the privacy that those scheme can guarantee.

4.2 Main Theorem

In the part, we prove Π_{aff} is parameter hiding, formally:

Theorem 7 (Main Theorem). *Assuming Π has $\mathcal{L}$-simulation-security where $\mathcal{L}$ is smoothed CLWW, then for any $\gamma = 2^{\omega(\log \lambda)}$, Π_{aff} is (γ, η, μ)-parameter hiding.*

Proof. According to the security notions, it is straightforward that if an ORE scheme is (γ, η, μ)-parameter hiding, then it is also (γ, η, μ)-scale hiding and (γ, η, μ)-shift hiding. Next we claim the converse proposition holds.

Claim. If an ORE scheme Π achieves (γ, η, μ)-scale hiding and (γ, η, μ)-shift hiding simultaneously, then Π is (γ, η, μ)-parameter hiding.

We sketch the proof by hybrid argument. For any $\gamma = 2^{\omega(\log \lambda)}$ and (η, μ)-smooth distribution D, firstly, by shift-hiding, there is no efficient adversary that distinguish (δ_0, ℓ_0) from $(\delta_0, 0)$ with non-negligible probability. Then due to scale-hiding, no efficient adversary can differ $(\delta_0, 0)$ from $(\delta_1, 0)$ with non-negligible probability. Thirdly, same as the first argument, any efficient adversary

can distinguish $(\delta_1, 0)$ from (δ_1, ℓ_1) with only negligible advantage. Combining together, Π achieves (γ, η, μ)-parameter hiding.

Thus, it suffices to show Π_{aff} is both (γ, η, μ)-scale hiding and (γ, η, μ)-shift hiding, due to space limit, we put the rigorous proof in our full version [12] .

5 ORE with smoothed CLWW Leakage

We start by defining the security we target via a smoothed CLWW leakage function. Then we recall a primitive for our construction called a *property-preserving hash (PPH) function*, and state and analyze our ORE construction using a PPH. In a later section we instantiate the PPH to complete the construction. Next, we give variant constructions with trade-offs between efficiency and leakage.

Now We define the non-adaptive version of the leakage profile for our construction. The leakage profile takes in input a vector of messages $\boldsymbol{m} = (m_1, \ldots, m_q)$ and produces the following:

$$\mathcal{L}_f(m_1, \ldots, m_q) := (\forall 1 \leq i, j, k \leq q, \mathbf{1}(m_i < m_j), \mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_i, m_k)))$$

By definition, it's easy to note that $\mathcal{L}_f$ leaks strictly less than CLWW. Except for the order of underlying plaintexts, it only leaks whether the position of $\mathsf{msdb}(m_i, m_j)$ and $\mathsf{msdb}(m_i, m_j)$ are the same, therefore the leakage profile preserve consistent if we left-shift all the plaintexts by one bit, which referring to $\mathcal{L}_f(\boldsymbol{m}) = \mathcal{L}_f(2\boldsymbol{m})$. Thus, $\mathcal{L}_f$ is smoothed CLWW.

5.1 Property Preserving Hash

Our construction will depend on a tool – *property preserving hash (PPH)*, which is essentially a property-preserving encryption scheme [35] without the decryption algorithm. In this section we recall the syntax and security of a PPH.

Definition 8. *A* property-preserving hash (PPH) scheme *is a tuple of algorithms $\Gamma = (\mathcal{K}_h, \mathcal{H}, \mathcal{T})$ with the following syntax:*

- *The key generation algorithm $\mathcal{K}_h$ is randomized, takes as input 1^λ and emits two outputs $(\mathsf{hk}, \mathsf{tk})$ that we refer to as the* hash key hk *and* test key tk*. These implicitly define a domain D and range R for the hash.*
- *The evaluation algorithm $\mathcal{H}$ is randomized, takes as input the hash key hk, an input $x \in D$, and emits a single output $h \in R$ that we refer to as the* hash of x*.*
- *The test algorithm $\mathcal{T}$ is deterministic, takes as input the test key tk and two hashes h_1, h_2, and emits a bit.*

Correctness of PPH schemes. Let P be a predicate on pairs of inputs. We define correctness of a PPH Γ with respect to P via the game $\mathrm{COR}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A})$, which is as follows: It starts by running $(\mathsf{hk}, \mathsf{tk}) \xleftarrow{\$} \mathcal{K}_h(1^\lambda)$ and gives tk to $\mathcal{A}$. Then $\mathcal{A}$ outputs x, y. The game computes $h \xleftarrow{\$} \mathcal{H}(\mathsf{hk}, x), h' \xleftarrow{\$} \mathcal{H}(\mathsf{hk}, y)$ and outputs 1 if

$\mathcal{T}(\mathsf{tk}, h, h') \neq P(x, y)$. We say that Γ is *computationally correct with respect to* P if for all efficient $\mathcal{A}$, $\Pr[\mathrm{COR}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A}) = 1]$ is a negligible function of λ.

Security of PPH schemes. We recall a simplified version of the security definition for PPH that is a weaker version of PPE security defined by Pandey and Rouselakis [35]. The definition is a sort of semantic security for random messages under chosen-plaintext attacks, except that the adversary is restricted from making certain queries.

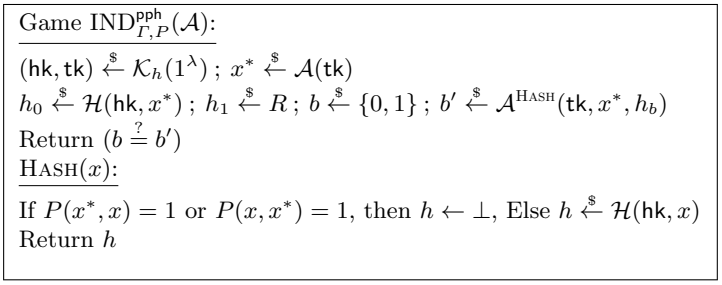

Fig. 6: Game $\mathrm{IND}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A})$.

Definition 9. *Let P be some predicate and $\Gamma = (\mathcal{K}_h, \mathcal{H}, \mathcal{T})$ be a PPH scheme with respect to P. For an adversary $\mathcal{A}$ we define the game $\mathrm{IND}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A})$ in Figure 6. The* restricted-chosen-input advantage of $\mathcal{A}$ *is defined to be* $\mathsf{Adv}^{\mathsf{pph}}_{\Gamma,P,\mathcal{A}}(\lambda) = 2\Pr[\mathrm{IND}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A}) = 1] - 1$. *We say that Γ is* restricted-chosen-input secure *if for all efficient adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{pph}}_{\Gamma,P,\mathcal{A}}(\lambda)$ is negligible.*

5.2 ORE from PPH

Construction. Let $F : K \times ([n] \times \{0,1\}^n) \to \{0,1\}^\lambda$ be a secure PRF. Let $P(x, y) = \mathbf{1}(x = y + 1)$ be the predicate that outputs 1 if and only if $x = y + 1$, and let $\Gamma = (\mathcal{K}_h, \mathcal{H}, \mathcal{T})$ be a PPH scheme with respect to P. In our construction, we interpret the output of F as a λ-bit integer, which is also the input domain of the PPH Γ. We define our ORE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ as follows:

- $\mathcal{K}(1^\lambda, M)$: On input the security parameter and message space $[M]$, the algorithm chooses a key k uniformly at random for F, and runs the key generation algorithm of the property preserving hash function $\Gamma.\mathcal{K}_h$ to obtain the hash and test keys $(\mathsf{hk}, \mathsf{tk})$. It sets $\mathsf{ck} \leftarrow \mathsf{tk}$, $\mathsf{sk} \leftarrow (k, \mathsf{hk})$ and outputs $(\mathsf{ck}, \mathsf{sk})$.
- $\mathcal{E}(\mathsf{sk}, m)$: On input the secret key sk and a message m, the algorithm writes the binary representation as m as $(b_1, \ldots, b_n)$, and then for $i = 1, \ldots, n$, it computes:

$$u_i = F(k, (i, b_1 b_2 \cdots b_{i-1} || 0^{n-i+1})) + b_i \mod 2^\lambda, \quad t_i = \Gamma.\mathcal{H}(\mathsf{hk}, u_i).$$

We note that u_i is computed by treating the PRF output as a member of $\{0, \ldots, 2^\lambda - 1\}$. Then it chooses a random permutation $\pi : [n] \to [n]$, and sets $v_i = t_{\pi(i)}$. The algorithm outputs $\mathsf{CT} = (v_1, \ldots, v_n)$.

- $\mathcal{C}(\mathsf{ck}, \mathsf{CT}_1, \mathsf{CT}_2)$: on input the public parameter, two ciphertexts $\mathsf{CT}_1, \mathsf{CT}_2$ where $\mathsf{CT}_1 = (v_1, \ldots, v_n), \mathsf{CT}_2 = (v'_1, \ldots, v'_n)$, the algorithm runs $\Gamma.\mathcal{T}(\mathsf{tk}, v_i, v'_j)$ and $\Gamma.\mathcal{T}(\mathsf{tk}, v'_i, v_j)$ for every $i, j \in [n]$. If there exists a pair (i^*, j^*) such that $\Gamma.\mathcal{T}(\mathsf{tk}, v_{i^*}, v'_{j^*}) = 1$, then the algorithm outputs 1, meaning $m_1 > m_2$; else if there exists a pair (i^*, j^*) such that $\Gamma.\mathcal{T}(\mathsf{tk}, v'_{i^*}, v_{j^*}) = 1$, then the algorithm outputs 0, meaning $m_1 < m_2$; otherwise it outputs $\perp$, meaning $m_1 = m_2$.

Correctness. For two messages m_1, m_2, let $(b_1, \ldots b_n)$ and $(b'_1, \ldots, b'_n)$ be their binary representations. Assuming $m_1 > m_2$, there must exists a unique index $i^* \in [n]$ such that $u_i = u'_i + 1$. Therefore correctness of Π is followed by correctness of PPH. We can use the same argument for the case $m_1 = m_2$ and $m_1 < m_2$. What is more interesting is its simulation based security, as it is the foundation for parameter hiding ORE, formally:

Theorem 10. *Assuming F is a secure PRF and Γ is restricted-chosen-input secure, Π is $\mathcal{L}_\mathsf{f}$-non-adaptively-simulation secure.*

Proof. We use a hybrid argument, and define a sequence of hybrid games as follows:

- H_{-1}: Real game $\mathrm{REAL}^{\mathsf{ore}}_{\Pi}(\mathcal{A})$;
- H_0: Same as H_{-1}, except replacing PRF $F_k(\cdot)$ by a truely random function F^* in the encryption oracle;
- $\mathsf{H}_{i\cdot q+j}$ Depend on a predicate $\mathsf{Switch}_{(i,j)}$ which is define below. If $\mathsf{Switch}_{(i,j)} = 0$, then $\mathsf{H}_{i\cdot q+j} = \mathsf{H}_{i\cdot q+j-1}$, else in procedure of $\mathcal{E}(m_j)$, u^j_i is replaced by a random string.

From the high level, we establish the proof by showing show that any adjacent hybrids are indistinguishable, and then we construct an efficient simulator S such that the output of H_{qn} and $\mathrm{SIM}^{\mathsf{ore}}_{\Pi,\mathcal{L}_\mathsf{f}}(\mathcal{A}, \mathcal{S})$ are statistically identical. For the predicate, we say $\mathsf{Switch}_{i,j} = 1$ if $\forall k \in [q], \mathsf{msdb}(m_j, m_k) \neq i$, and 0 otherwise. We note that when $\mathsf{Switch}_{i,j} = 0$, there exists u^k_i such that $u^j_i = u^k_i \pm 1$, the relation which can be detected by the test algorithm of PPH(for the i-th bit of m_j, we call such a bit a leaky bit), which means we cannot replace it with random string, otherwise adversary can trivially distinguish it. In the following we firstly prove any adjacent objects are computational indistinguishable.

Lemma 11. *Assuming Γ is restricted-chosen-input secure, for any $k \in [qn]$ $\mathsf{H}_{k-1} \overset{\mathsf{comp}}{\approx} \mathsf{H}_k$.*

Proof. Due to the security of PRF, it's trivial that $\mathsf{H}_{-1} \overset{\mathsf{comp}}{\approx} \mathsf{H}_0$, and for any $k > 0$(for ease, $k = i^* \cdot q + j^*$ where $i^* \in [n-1], j^* \in [q]$), it suffices to show $\mathsf{H}_{k-1} \overset{\mathsf{comp}}{\approx} \mathsf{H}_k$ under the condition $\mathsf{Switch}_{i^*,j^*} = 1$($\mathsf{Switch}_{i^*,j^*} = 0$ implies $\mathsf{H}_{k-1} = \mathsf{H}_k$). We prove that if there exists adversary $\mathcal{A}$ that distinguish H_k from

$\mathsf{H}_{\mathsf{k}-1}$ with noticeable advantage ϵ, then we can construct a simulator $\mathcal{B}$ wins the restricted-chosen-input game with ϵ-negl. Here is the description of $\mathcal{B}$. Firstly it runs $\mathrm{IND}_{\Gamma}^{\mathsf{pph}}$, and sends tk as the comparison key ck to $\mathcal{A}$. After receiving a sequence of plaintext $m_1, \ldots, m_q$, it picks a random function F^*(using the lazy sampling technique for instance), sets $X^* = F^*(i^*, b_1^{j^*} b_2^{j^*} \cdots b_{i^*-1}^{j^*} || 0^{n-i^*+1}) + b_{i^*}^{j^*}$ where b_i^j is the i-th bit of m_j. Then it sends X^* to its challenger in restricted-chosen-input game and gets back T as the challenge term. To simulate the encryption oracle, $\mathcal{B}$ works as follows:

1. $(i', j') > (i^*, j^*)$(here using a natural order for tuples, $(i, j) > (i', j')$ iff $iq + j > i'q + j'$), $\mathcal{B}$ computes:

$$u_{i'}^{j'} = F^*(i^*, b_1^{j'} b_2^{j'} \cdots b_{i'-1}^{j'} || 0^{n-i'+1}) + b_{i'}^{j'}; t_{i'}^{j'} = \Gamma.\mathcal{H}(\mathsf{hk}, u_{i'}^{j'})$$

2. $(i', j') < (i^*, j^*) \cap \mathsf{Switch}_{i',j'} = 0$, then same as above, else $u_{i'}^{j'} \xleftarrow{\$} \{0,1\}^{\lambda}, t_{i'}^{j'} = \Gamma.\mathcal{H}(\mathsf{hk}, u_{i'}^{j'})$.
3. sets $t_{i^*}^{j^*} = T$, and $\forall j \in [q]$, picks a random permutation π_j and outputs the ciphertexts $\mathsf{CT}_j = (t_{\pi_j(1)}^j, \ldots, t_{\pi_j(n)}^j)$.

Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs[11].

Since F^* is a random function, $\Pr[u_{i'}^{j'} = X^* \pm 1]$ is negligible for all $(i', j') \neq (i^*, j^*)$, which means $\mathcal{B}$ fails to simulate the encryption oracle with only negligible probability. Besides, when $T = \Gamma.\mathcal{H}(\mathsf{hk}, X^*)$, $\mathcal{B}$ properly simulates H_{k-1}, and if T is random, then $\mathcal{B}$ simulates H_k(due to the PRF security, the distribution of $\Gamma.\mathcal{H}(\mathsf{hk}, r) : r \xleftarrow{\$} \{0,1\}^{\lambda}$ is computationally close to a random variable that uniformly sampled from the range of Γ). Hence, if $\mathbf{Adv}(\mathcal{A})$ is noticeable, then $\mathcal{B}$'s advantage is also noticeable. □

In the following, we describe an efficient simulator S such that the output of H_{qn} and $\mathrm{SIM}_{\Pi,\mathcal{L}_{\mathsf{f}}}^{\mathsf{ore}}(\mathcal{A}, \mathcal{S})$ are statistically identical. Roughly speaking, we note that $\mathsf{Switch}_{i,j} = 1$ means that i-th bit of m_j is not a leaky bit, indicating that its value would not affect the leakage profile whp. Hence, it suffices to only simulate the leaky bit of each individual message, which can be extracted by $\mathcal{L}_f$, and sets the rest just as random string. Due to the final random permutations, H_{qn} and $\mathrm{SIM}_{\Pi,\mathcal{L}_{\mathsf{f}}}^{\mathsf{ore}}(\mathcal{A}, \mathcal{S})$ are statistically identical. Formally:

Description of the simulator. For fixed a message set $\mathcal{M} = \{m_1, \ldots, m_q\}$ (without loss of generality, we assume $m_1 > \ldots > m_q$), the simulator $\mathcal{S}$ is given the leakage information $\mathcal{L}_f(m_1, \ldots, m_q)$. $\mathcal{S}$ firstly keeps a $q \times n$ matrix $\mathcal{B}$ and runs a recursive algorithm $\mathsf{FillMatrix}(1, 1, q)$ to fill in the entries, as follows:

- If $j = k$, then $\forall i' \in [i, n]$, $\mathcal{B}[j][i'] = r$ where $r \xleftarrow{\$} \{0,1\}^{\lambda}$;
- Else, it proceeds as follows:
  - searches the smallest $j^* \in [j, k]$ s.t. $P(m_j, m_{j^*}) = P(m_j, m_k)$;

[11] We note that $\mathcal{B}$ does not have hk, what it does is to call the hash oracle

- sets $\mathcal{B}[j'][i] = r', \forall j' \in [j, j^* - 1]; \mathcal{B}[j'][i] = r' - 1, \forall j' \in [j^*, k]$, where $r' \xleftarrow{\$} \{0,1\}^\lambda$;
- runs $\mathsf{FillMatrix}(i+1, j, j'-1)$ and $\mathsf{FillMatrix}(i+1, j', k)$ recursively.

More concretely, our recursive algorithm is to fill in the entries by

$$\mathsf{FillMatrix}(i, j, k),\ \forall i \in [n], j \leq k \in [q]$$

Then $\mathcal{S}$ runs $\Gamma.\mathcal{K}_h(1^\lambda)$ and gets the keys tk, hk, and sets $t_{i,j} = \Gamma.\mathcal{H}(\mathsf{hk}, \mathcal{B}[j][i])$, $\forall i \in [n], j \in [q]$. Finally, $\mathcal{S}$ samples random permutations π_j, outputs CT_j as $\mathsf{CT}_j = (t^j_{\pi_j(1)}, \ldots, t^j_{\pi_j(n)})$ We note that the FillMatrix algorithm terminates after at most qn steps as each cell will not be written twice, hence $\mathcal{S}$ is an efficient simulator.

Finally we claim that $\mathcal{S}$ properly simulates the relevant games. We first observe that the simulator identifies how many leaked bits (prefixes) there are for the messages $m_1, \ldots, m_q$. Recall that if messages $m_1, \ldots, m_q$ share the same prefix up to the $\ell-1$-th bit, and if there exists (the first) i^* such that $\mathsf{msdb}(m_1, m_{i^*}) = \mathsf{msdb}(m_1, m_q)$, then we can conclude that $\{m_1, \ldots, m_{i^*-1}\}$ has 1 on their ℓ-th bit, and and $\{m_{i^*}, \ldots, m_q\}$ has 0 on their ℓ-th bit. This way the ℓ-th bit of these messages are leaked. The simulator recursively identifies other leaked bits for these two sets. At the end, for each message, how many prefixes whose next bits are leaked will be identified. As this information will also be identified in the hybrid H_{qn}. So a random permutation (for H_{qn} and the simulation) will hide these leaked prefixes, except the total number. Thus, our simulation is identical to H_{qn}, and we establish the entire proof. □

5.3 More efficient comparisons

The construction above needs to run $O(n^2)$ times PPH test algorithm for one single comparison, which is very expensive for real application. In this part, we present a variant ORE achieving better efficiency but with a weaker leakage profile, which only requires $O(n)$ pairings in each individual comparison. And what's more interesting is that this weaker leakage profile is also smoothed CLWW, that means we can still construct a parameter hiding ORE based on it, along with better efficiency. From the high level, we fix a permutation for all encryptions(this permutation is part of the secret key now), rather than sampling fresh permutation for each ciphertext. Therefore, in the comparison, we only need run the PPH test for pairs that share the same index, which means only $O(n)$ pairings for one comparison. Formally:

Construction. Let F be a secure PRF with the same syntax as above, let $P(x, y) = \mathbf{1}(x = y + 1)$ be the relation predicate that outputs 1 if and only if $x = y+1$, and let $\Gamma = (\mathcal{K}_h, \mathcal{H}, \mathcal{T})$ be a PPH scheme with respect to P, as before. We define our ORE scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ as follows:

– $\mathcal{K}(1^\lambda, M)$: On input the security parameter and message space $[M]$, the algorithm chooses a key k uniformly at random for F, runs $\Gamma.\mathcal{K}_h$ to obtain the

hash and test keys $(\mathsf{hk}, \mathsf{tk})$, and samples a random permutation $\pi : [n] \to [n]$. It sets $\mathsf{ck} \leftarrow \mathsf{tk}$, $\mathsf{sk} \leftarrow (k, \mathsf{hk}, \pi)$ and outputs $(\mathsf{ck}, \mathsf{sk})$.

- $\mathcal{E}(\mathsf{sk}, m)$: On input the secret key SK and a message m, the algorithm computes the binary representation of $m = (b_1, \ldots, b_n)$, and then calculates:

$$u_i = F(k, (i, b_1 b_2 \cdots b_{i-1} || 0^{n-i+1})) + b_i, \quad t_i = \Gamma.\mathcal{H}(\mathsf{hk}, u_i).$$

  Then it sets $v_i = t_{\pi(i)}$ and outputs $\mathsf{CT} = (v_1, \ldots, v_n)$.
- $\mathcal{C}(\mathsf{ck}, \mathsf{CT_1}, \mathsf{CT_2})$: on input the public parameter, two ciphertexts $\mathsf{CT_1}, \mathsf{CT_2}$ where $\mathsf{CT_1} = (v_1, \ldots, v_n), \mathsf{CT_2} = (v'_1, \ldots, v'_n)$, the algorithm runs $\Gamma.\mathcal{T}(\mathsf{tk}, v_i, v'_i)$ for every $i \in [n]$. If there exists i^* such that $\Gamma.\mathcal{T}(\mathsf{tk}, v_{i^*}, v'_{i^*}) = 1$, then the algorithm outputs 1, meaning $m_1 > m_2$; else if there exists a pair i^* such that $\Gamma.\mathcal{T}(\mathsf{tk}, v'_{i^*}, v_{i^*}) = 1$, then the algorithm outputs 0, meaning $m_1 < m_2$; otherwise it outputs it outputs $\perp$, meaning $m_1 = m_2$.

Now, we give the description of the leakage profile, which takes $\boldsymbol{m} = \{m_1, \ldots, m_q\}$ as input and produces:

$$\mathcal{L}'_f(m_1, \ldots, m_q) := (\forall 1 \le i, j, k, l \le q, \mathbf{1}(m_i < m_j), \mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_k, m_l)))$$

Compared to $\mathcal{L}_f$, $\mathcal{L}'_f$ gives extra information that $\mathbf{1}(\mathsf{msdb}(m_i, m_j) = \mathsf{msdb}(m_k, m_l))$ even when $i \neq k$. However, $\mathcal{L}'_f$ is still strictly stronger than CLWW, and for any $\boldsymbol{m}$, it's obvious that $\mathcal{L}'_f(\boldsymbol{m}) = \mathcal{L}'_f(2\boldsymbol{m})$, which gives evidence that $\mathcal{L}'_f$ is also smoothed CLWW. And for its simulation based security, applying exactly the same argument as the proof of Theorem 10, we can establish the following theorem.

Theorem 12. *The ORE scheme Π is $\mathcal{L}'_f$-non-adaptive-simulation secure, assuming F is a secure PRF and Γ is restricted-chosen-input secure.*

Therefore, to achieve the privacy of parameter hiding, we can use this efficient scheme as an alternative, such that we only need $O(n)$ pairings for each comparison.

6 PPH from Bilinear Maps

We construct a PPH scheme for the predicate P required in our ORE construction. That is, $P(x, y) = 1$ if and only if $x = y + 1$.

We let $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{Z}_p$ be a PRF, where p is a prime to be determined at key generation.

Construction. We now define our PPH $\Gamma = (\mathcal{K}_h, \mathcal{H}, \mathcal{T})$.

- $\mathcal{K}_h(1^\lambda)$ This algorithm takes the security parameter as input. It samples descriptions of prime-order p groups $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$, generators $g \in \mathbb{G}, \hat{g} \in \hat{\mathbb{G}}$, a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$. It then chooses $k \xleftarrow{\$} \{0,1\}^\lambda$. It sets the hash key $\mathsf{hk} \leftarrow (k, g, \hat{g})$, the test key $\mathsf{tk} \leftarrow (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$, a description of the bilinear map and groups, and outputs $(\mathsf{hk}, \mathsf{tk})$.

– $\mathcal{H}(\mathsf{hk}, x)$ This algorithm takes as input the hash key hk, an input x, picks two random non-zero $r_1, r_2 \in \mathbb{Z}_p$ and outputs

$$\mathcal{H}(\mathsf{hk}, x) = (g^{r_1}, g^{r_1 \cdot F(k,x)}, \hat{g}^{r_2}, \hat{g}^{r_2 \cdot F(k,x+1)}).$$

– $\mathcal{T}(\mathsf{tk}, h_1, h_2)$ To test two hash values (A_1, A_2, B_1, B_2) and (C_1, C_2, D_1, D_2), $\mathcal{T}$ outputs 1 if

$$e(A_1, D_2) = e(A_2, D_1),$$

and otherwise it outputs 0.

Hence the domain D is $\{0,1\}^\lambda$ and the range R is $(\mathbb{G}^2, \hat{\mathbb{G}}^2)$

Correctness. Correctness reduces to testing if $F(k, y+1) = F(k, x)$. If $x = y+1$ then this always holds. If not, then it is easily shown that finding x, y with this property (and without knowing the key) with non-negligible probability leads to an adversary that contradicts the assumption that F is a PRF.

Security. We prove that PPH is restricted-chosen-input secure, assuming that F is a PRF and that the following assumption holds.

Definition 13. *Let $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ be prime-order p groups, g be generator of $\mathbb{G}$ and $\hat{g}$ be a generator of $\hat{\mathbb{G}}$, tand $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$ be a bilinear pairing. We say the* symmetric external Diffie-Hellman assumption holds *with respect to these groups and pairing if for all efficient $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] \Pr[\mathcal{A}(g, g^a, g^b, T) = 1]|$$

and

$$|\Pr[\mathcal{A}(\hat{g}, \hat{g}^a, \hat{g}^b, \hat{g}^{ab}) = 1] \Pr[\mathcal{A}(\hat{g}, \hat{g}^a, \hat{g}^b, T) = 1]|$$

are negligible functions of λ, where a, b, c are uniform over $\mathbb{Z}_p$ and T is uniform over G_T.

We can now state and prove our security theorem.

Theorem 14. *Our PPH Γ is restricted-chosen-input secure, assuming F is a PRF and the SXDH assumption hold with respect to the appropriate groups and pairing.*

Proof. We use a hybrid argument. Let $(A_1, A_2, B_1, B_2) \in \mathbb{G}^2 \times \hat{\mathbb{G}}^2$ denote the challenge hash value given to the adversary during the real game $\mathsf{H}_0 = \mathrm{IND}^{\mathsf{pph}}_{\Gamma,P}(\mathcal{A})$. Additionally, let R be a random element of $\mathbb{G}$, $\hat{R}$ be a random element of $\hat{\mathbb{G}}$, both independent of the rest of the random variables under consideration. Then we define the following hybrid experiments:

– H_1: At the start of the game, a uniformly random function $F^* \xleftarrow{R} \mathsf{Funs}[\{0,1\}^\lambda, \{0,1\}^\lambda]$ is sampled instead of the PRF key K, the rest remain unchanged.
– H_2: The challenge hash value is (A_1, R, B_1, B_2), where $R \xleftarrow{\$} \mathbb{G}$.
– H_3: The challenge hash value is $(A_1, R, B_1, \hat{R})$, where $R \xleftarrow{\$} \hat{\mathbb{G}}$.

In H_3, the adversary is given a random element from the range $\mathcal{R}$. Therefore,

$$\mathsf{Adv}^{\mathsf{pph}}_{\Gamma,P,\mathcal{A}}(\lambda) = |Pr[\mathsf{H}_0 = 1] - Pr[\mathsf{H}_3 = 1]|$$

To prove H_0 is indistinguishable from H_3, we show that each step of the hybrid is indistinguishable from the next. First, it is apparent that H_0 and H_1 are computational indistinguishable by the PRF security, then:

Lemma 15. $\mathsf{H}_1 \approx \mathsf{H}_2$ *under the SXDH assumption.*

Let $\mathcal{A}$ be an adversary playing the PPH security game, and let

$$\epsilon = |Pr[\mathsf{H}_1 = 1] - Pr[\mathsf{H}_2 = 1]|.$$

Then we can build adversary $\mathcal{B}$ that solves SXDH with advantage ϵ. $\mathcal{B}$ is given as input $(g, \hat{g}, B, C)$ and the challenge term T. $\mathcal{B}$ works as follows:

- $\mathcal{B}$ sets $\mathsf{tk} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, e)$ and sends it to $\mathcal{A}$. After receiving $x^* \xleftarrow{\$} \mathcal{A}(\mathsf{tk})$ it simulates a random function F^* via lazy sampling, and it will implicitly set $F^*(x^*) = b$, the discrete logarithm of B. It prepares the challenge as by selecting $r^* \xleftarrow{\$} \mathbb{Z}_p$ and computing

  $$A_1 = g^c, A_2 = T, B_1 = \hat{g}^{r^*}, B_2 = \hat{g}^{r^* F^*(x^*+1)}$$

  and runs $\mathcal{A}$ on input $\mathsf{tk}, x^*, (A_1, A_2, B_1, B_2)$.
- To answer hash query for $x \neq x^*$ from $\mathcal{A}$, $\mathcal{B}$ calculates $F^*(x)$ and $F^*(x+1)$ (note that $x, x+1 \neq x^*$). Then $\mathcal{B}$ picks r_1, r_2 randomly and computes:

  $$\mathcal{H}(x) = g^{r_1}, g^{r_1 \cdot F^*(x)}, \hat{g}^{r_2}, \hat{g}^{r_2 \cdot F^*(x+1)};$$

  If $\mathcal{A}$ queries $x = x^*$, $\mathcal{B}$ calculates $F^*(x^*+1)$, picks $r'_1, r'_2 \xleftarrow{\$} \mathbb{Z}_p$, and computes

  $$\mathcal{H}(x^*) = g^{r'_1}, B^{r'_1}, \hat{g}^{r'_2}, \hat{g}^{r'_2 \cdot F^*(x^*+1)};$$

- Finally $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

We note that in $\mathcal{A}$'s view, without querying $\mathcal{A}(x^* - 1)$, $\mathcal{B}$ simulates the game properly. If $T = g^{bc}$, then $\mathcal{B}$ simulates H_1, and if T s random then it simulates H_2. Hence if $\mathcal{A}$ has an advantage ϵ in distinguishing H_1 and H_2, then $\mathcal{B}$ has the same advantage to break SXDH assumption.

We also have the following lemma:

Lemma 16. $\mathsf{H}_2 \approx \mathsf{H}_3$ *under the SXDH assumption.*

The proof is exactly the same as the prior hybrid step, except in the group $\hat{\mathbb{G}}$ part of the hash instead of $\mathbb{G}$. We omit the details.

Collecting the steps completes the proof of Theorem 14.

Acknowledgments

David Cash is supported by NSF CNS-1453132. Feng-Hao Liu is supported by NSF CNS-1657040. Adam O'Neill is supported in part by NSF CNS-1650419. Mark Zhandry is supported by NSF. David Cash and Cong Zhang are partially supported by DARPA and SSC Pacific under contract N66001-15-C-4070. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF, DARPA or SSC Pacific.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *SIGMOD*, 2004.
2. A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal security with cipherbase. In *CIDR*, 2013.
3. A. Arasu, K. Eguro, R. Kaushik, and R. Ramamurthy. Querying encrypted data (tutorial). In *ICDE*, 2013.
4. S. Bajaj and R. Sion. Trusteddb: A trusted hardware-based database with privacy and data confidentiality. *TKDE*, 26(3):752–765, 2014.
5. M. Bellare, A. Boldyreva, and A. ONeill. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference*, pages 535–552. Springer, 2007.
6. A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 224–241. Springer, 2009.
7. A. Boldyreva, N. Chenette, and A. ONeill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*, pages 578–595. Springer, 2011.
8. D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 563–594. Springer, 2015.
9. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Annual International Cryptology Conference*, pages 455–469. Springer, 1997.
10. D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *CCS*, 2015.
11. D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology–CRYPTO 2013*, pages 353–373. Springer, 2013.
12. D. Cash, F.-H. Liu, A. O'Neill, M. Zhandry, and C. Zhang. Parameter-hiding order revealing encryption. Cryptology ePrint Archive, Report 2018/698, 2018. `https://eprint.iacr.org/2018/698`.
13. Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *ACNS*, 2005.
14. M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 577–594. Springer, 2010.

15. N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, 2016.
16. J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the multilinear map over the integers. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 3–12, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.
17. J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Cryptanalysis of ggh15 multilinear maps. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 607–628, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.
18. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *CCS*, 2006.
19. J. L. Dautrich Jr and C. V. Ravishankar. Compromising privacy in precise query protocols. In *EDBT*, 2013.
20. F. B. Durak, T. M. DuBuisson, and D. Cash. What else is revealed by order-revealing encryption? In *ACM CCS*, 2016.
21. E.-J. Goh et al. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.
22. P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. Cryptology ePrint Archive, Report 2016/895, 2016. `http://eprint.iacr.org/2016/895`.
23. H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 216–227, New York, NY, USA, 2002. ACM.
24. B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. Secure multidimensional range queries over outsourced data. *VLDBJ*, 21(3):333–358, 2012.
25. M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, 2012.
26. M. S. Islam, M. Kuzu, and M. Kantarcioglu. Inference attack against encrypted range queries on outsourced databases. In *CODASPY*, 2014.
27. M. Joye and A. Passelègue. Function-revealing encryption. 2016.
28. S. Kamara and T. Moataz. Sql on structurally-encrypted databases. Cryptology ePrint Archive, Report 2016/453, 2016. `http://eprint.iacr.org/`.
29. K. Lewi and D. J. Wu. Order-revealing encryption: New constructions, applications, and lower bounds. In *ACM CCS*, 2016.
30. C. Liu, L. Zhu, M. Wang, and Y.-a. Tan. Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, 265:176–188, 2014.
31. A. Mandal and A. Roy. Relational hash: Probabilistic hash for verifying relations, secure against forgery and more. In *Annual Cryptology Conference*, pages 518–537. Springer, 2015.
32. E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over ggh13. In *Proceedings, Part II, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9815*, pages 629–658, Berlin, Heidelberg, 2016. Springer-Verlag.
33. P. Mohassel and M. Franklin. Efficiency tradeoffs for malicious two-party computation. In *International Workshop on Public Key Cryptography*, pages 458–473. Springer, 2006.
34. M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *CCS*, 2015.

35. O. Pandey and Y. Rouselakis. Property preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 375–391. Springer, 2012.
36. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *SOSP*, 2011.
37. D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP*, 2000.