Near-Optimal Oblivious Key-Value Stores for Efficient PSI, PSU and Volume-Hiding Multi-Maps

Alexander Bienstock, *New York University;* Sarvar Patel and Joon Young Seo, *Google;* Kevin Yeo, *Google and Columbia University*

https://www.usenix.org/conference/usenixsecurity23/presentation/bienstock

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

Near-Optimal Oblivious Key-Value Stores for Efficient PSI, PSU and Volume-Hiding Multi-Maps

Alexander Bienstock[*] *Sarvar Patel*[†] *Joon Young Seo*[‡] *Kevin Yeo*[§]

Abstract

In this paper, we study oblivious key-value stores (OKVS) that enable encoding n key-value pairs into length m encodings while hiding the input keys. The goal is to obtain high rate, n/m, with efficient encoding and decoding algorithms. We present RB-OKVS built on random band matrices that obtains near-optimal rates as high as 0.97 whereas prior works could only achieve rates up to 0.81 with similar encoding times.

Using RB-OKVS, we obtain state-of-the-art protocols for private set intersection (PSI) and union (PSU). Our semi-honest PSI has up to 12% smaller communication and 13% reductions in monetary cost with slightly larger computation. We also obtain similar improvements for both malicious and circuit PSI. For PSU, our protocol obtains improvements of up to 22% in communication, 40% in computation and 21% in monetary cost. In general, we obtain the most communication- and cost-efficient protocols for all the above primitives.

Finally, we present the first connection between OKVS and volume-hiding encrypted multi-maps (VH-EMM) where the goal is to outsource storage of multi-maps while hiding the number of values associated with each key (i.e., volume). We present RB-MM with 16% smaller storage, 5x faster queries and 8x faster setup than prior works.

1 Introduction

In recent years, there is a growing interest in enabling independent parties to collaborate and jointly perform computation to gain insights into their combined data. For many settings, the data sets held by each party contain sensitive information and must be kept private. Therefore, the goal is to ensure that each party only learns the desired, predetermined outputs and nothing else. Several organizations such as Google [3], Meta [5] and Signal [1] actively work on these problems.

One important problem is computing the intersection between two data sets, denoted by private set intersection (PSI). PSI is an important problem due to its numerous applications to real-world problems such as ads attribution [41], contact discovery [28, 44, 47], contact tracing [31] and password leak detection [67] to list some examples. In PSI, there are two parties that hold sets of identifiers X and Y respectively with the goal of computing the intersection $X \cap Y$ (i.e., elements appearing in both X and Y). For privacy, each party should learn no other information beyond the intersection, $X \cap Y$, and the size of the other party's set.

Another key problem is private set union (PSU) that considers the same setting as PSI with two parties holding sets X and Y respectively with the goal of computing their union, $X \cup Y$. Privacy remains identical where each party should learn only the output $X \cup Y$ and nothing else except for the size of the other party's set. PSU is an essential component for multiple applications including aggregation of network events [15], improving blocklist accuracy [62], security risk assessments [38] and universal identifier generation [14].

It turns out that both PSI and PSU require the usage of a similar primitive known as oblivious key-value stores (OKVS). Garimella *et al.* [34] first observed that many prior PSI protocols (such as [25, 30, 50, 56, 58, 60]) implicitly relied upon similar properties. They abstracted out these properties and defined them as an OKVS. Furthermore, recent PSI works [34, 61] explicitly build PSI using OKVS. The usage of OKVS extends to other variants of PSI including circuit PSI [19, 58, 64] and multi-party PSI [10, 18, 40, 50, 53, 71] for more than two parties. Additionally, recent PSU protocols also heavily rely upon OKVS constructions [51, 70].

In both PSI and PSU, the underlying OKVS constitutes a significant portion of the total communication and computation costs. Therefore, it is important to study OKVS as any improved OKVS constructions would have an immediate impact on the efficiency of PSI and PSU. In this work, we present novel and improved OKVS schemes and uncover new

[*] New York University, abienstock@cs.nyu.edu. Part of work done while interning at Google.
[†] Google, sarvar@google.com.
[‡] Google, jyseo@google.com.
[§] Google and Columbia University, kwlyeo@google.com.
The full version of this paper may be found at [11].

	Rate	Encoding	Decoding
Polynomial	1	$O(n \log^2 n)$	$O(n)$
Random Matrix [34]	1	$O(n^3)$	$O(n)$
Bloom Filter [30]	$O(1/\lambda)$	$O(n\lambda)$	$O(\lambda)$
PaXoS [56]	0.4	$O(n\lambda)$	$O(\lambda)$
3H-GCT [34]	0.77-0.81	$O(n\lambda)$	$O(\lambda)$
RR22 [61]	0.78-0.81	$O(n\lambda)$	$O(\lambda)$
Ours: RB-OKVS	**0.91-0.97**	$O(n\lambda)$	$O(\lambda)$

Figure 1: OKVS constructions with error probability $2^{-\lambda}$ and costs for encoding n key-value pairs and decoding one value.

applications of OKVS including volume-hiding multi-maps.

1.1 Our Contributions

Oblivious Key-Value Stores (OKVS). As our main contribution, we present a novel OKVS construction, RB-OKVS. At its core, RB-OKVS is built on top of a framework that embeds the input key-value pairs in an efficiently solvable system of linear equations defined by a family of random matrices. In particular, RB-OKVS relies upon random band matrices [29] where each row consists of a single w-bit band of uniformly random bits. We note that RB-OKVS significantly deviates from prior OKVS constructions built using novel modifications of cuckoo hashing [34, 56, 61]. We point readers to Section 3 for the description of RB-OKVS.

RB-OKVS encodes n key-value pairs into encodings of size as small as $1.03n$, obtaining nearly optimal rates of 0.97. Prior works [34, 61] only obtained rates of 0.81 when restricted to $O(n\lambda)$ encoding time and $2^{-\lambda}$ error probability. Furthermore, we show that RB-OKVS is highly parameterizable, enabling trade-offs between the rate and encoding times. For a variety of rates better than all prior OKVS schemes, RB-OKVS still has the fastest encoding times. We present a comparison with prior OKVS schemes in Figure 1 and point readers to Section 6.1 for experimental evaluation and comparisons.

Private Set Intersection (PSI). By plugging RB-OKVS into known PSI frameworks, we immediately obtain improved constructions for semi-honest, malicious and circuit PSI over prior state-of-the-art [61]. For semi-honest, our protocol with RB-OKVS has 12% reductions in communication and 13% reductions in monetary cost in exchange for slightly larger latencies. We also obtain 10% less communication and 11% smaller costs for malicious PSI using RB-OKVS. Our circuit PSI also enjoys 12% smaller communication and 9% less monetary cost with slightly more computation. For all three variants, our PSI protocols also obtain the fastest latencies when considering more network constrained settings due to the smaller communication requirements (see Section 6.2).

Private Set Union (PSU). A recent work by Zhang *et al.* [70] presented a new PSU protocol with linear communication and computation costs. As a core component, this PSU protocol utilizes a slightly modified version of the 3H-GCT OKVS [34]. We show that, by plugging in RB-OKVS, we can obtain a PSU protocol with up to 22% smaller communication and 37% less computation (see Section 6.3).

Volume-Hiding Encrypted Multi-Maps (VH-EMM). Finally, we explore the utility of OKVS beyond PSI and PSU. We show the first connection between OKVS and VH-EMM. A VH-EMM enables outsourcing an encrypted multi-map to an untrusted server without revealing the number of values (volume) corresponding to any key. Multiple important applications rely upon VH-EMM including searchable encryption [66] and encrypted databases [45]. We present RB-MM that is built directly from RB-OKVS. Compared to state-of-the-art constructions [69], RB-MM uses 16% less storage, has 5x smaller query times and 8x smaller setup times while maintaining optimal communication (see Section 5).

2 Preliminaries

Throughout the paper, we will denote any vector $\mathbf{v}$ as column vectors and its transpose $\mathbf{v}^\intercal$ as row vectors. Vectors written as $[x_1, \ldots, x_n]$ will be row vectors and its transpose $[x_1, \ldots, x_n]^\intercal$ will be column vectors. For two equal-length vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, we denote the dot product as $\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} \mathbf{u}[i] \cdot \mathbf{v}[i]$. We denote a $n \times m$ matrix with n rows and m columns as $\mathbf{M} \in \mathbb{F}^{n \times m}$. For a matrix $\mathbf{M} \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{v} \in \mathbb{F}^m$, we denote the matrix-vector product as $\mathbf{M} \cdot \mathbf{v} = [\mathbf{M}[1] \cdot \mathbf{v}, \ldots, \mathbf{M}[n] \cdot \mathbf{v}]^\intercal$ where $\mathbf{M}[i] \in \mathbb{F}^m$ is the i-th row of $\mathbf{M}$. We say that we solve the linear system corresponding to matrix $\mathbf{M} \in \mathbb{F}^{n \times m}$ and vector $\mathbf{u} \in \mathbb{F}^n$ if we can find a vector $\mathbf{v} \in \mathbb{F}^m$ such that $\mathbf{M} \cdot \mathbf{v} = \mathbf{u}$.

2.1 Oblivious Key-Value Stores (OKVS)

We define the notion of oblivous key-value stores (OKVS) introduced by Garimella *et al.* [34]. At a high level, an OKVS enables encoding n pairs of key-value pairs such that an adversary is unable to reverse engineer the original input keys when given the encoding, assuming the input values are random. In other words, the encoding is *oblivious* to the input keys.

Definition 1 (Oblivious Key-Value Store). *An oblivious key-value store (OKVS) is parameterized by a key universe $\mathcal{K}$ and value universe $\mathcal{V}$ and consists of the two functions:*

- $\mathbf{S} \leftarrow \mathsf{Encode}(I; R)$*: The encode algorithm receives a set of n key-value pairs $I = \{(k_1, v_1), \ldots, (k_n, v_n)\} \in (\mathcal{K} \times \mathcal{V})^n$ with n distinct keys and randomness R and outputs the encoding $\mathbf{S} \in \mathcal{V}^m \cup \{\perp\}$.*
- $v \leftarrow \mathsf{Decode}(\mathbf{S}, k; R)$*: The decode algorithm receives the encoding $\mathbf{S} \in \mathcal{V}^m$, a key $k \in \mathcal{K}$ and randomness R and outputs the associated value $v \in \mathcal{V}$.*

An OKVS has error probability ε if, for all sets of n key-value pairs $I = \{(k_1, v_1), \ldots, (k_n, v_n)\} \subseteq (\mathcal{K} \times \mathcal{V})^n$ with n

distinct keys, the following holds for all $i \in [n]$:

$$\Pr[\mathsf{Decode}(\mathbf{S}, k_i) \neq v_i \mid \mathbf{S} \leftarrow \mathsf{Encode}(I)] \leq \varepsilon.$$

An OKVS is computationally oblivious, if for all pairs of sets of n distinct keys $A = \{k_1, \ldots, k_n\} \subseteq \mathcal{K}$ and $B = \{k'_1, \ldots, k'_n\} \subseteq \mathcal{K}$ and n values $v_1, \ldots, v_n$ each drawn uniformly at random from $\mathcal{V}$, then a computational adversary cannot distinguish between the following two encodings:

1. $\mathbf{S} \leftarrow \mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$.
2. $\mathbf{S}' \leftarrow \mathsf{Encode}(\{(k'_1, v_1), \ldots, (k'_n, v_n)\})$.

As a note, there are no guarantees for the case of executing Decode on a key k that is not an input key.

Efficiency Measures. When evaluating the efficiency of an OKVS, there are typically three important measures: rate, encoding cost and decoding cost. The rate is computed as the number of key-value pairs n divided by the size of the encoding m, that is, n/m. The best possible rate is 1 when the encoding has optimal size n. The encoding cost is the computational overhead required to encode an input of n key-value pairs and the decoding cost is the computational overhead required to decode the value associated to a single key.

Additional Properties. On top of correctness and obliviousness, it is convenient in many applications such as PSI and PSU for an OKVS to satisfy additional properties. We will build our OKVS construction to satisfy all these properties to enable wide applicability to various problems.

The first property of *linearity* concerns the structure of the decode algorithm that was shown to be useful for PSI [34]. An OKVS is linear if the decode algorithm is a linear combination of a subset of the encoding entries.

Definition 2 (Linear)**.** *An OKVS is* linear *if there exists a function $d : \mathcal{K} \to \mathcal{V}^m$ such that for all $k \in \mathcal{K}$ and $\mathbf{S} \in \mathcal{V}^m$*

$$\mathsf{Decode}(\mathbf{S}, k) = \sum_{i=1}^{m} d(k)[i] \cdot \mathbf{S}[i].$$

A special case of linearity is a binary OKVS [34] that is useful for certain PSI protocols. In a binary OKVS, the decode algorithm is just the sum of a subset of encoding entries (the function d maps to a binary string, $d : \mathcal{K} \to \{0,1\}^m$). We do not rely on the binary property in our paper, but note that our OKVS satisfies the property that may be useful in the future.

The next property is a strengthening of security denoted as being doubly oblivious where the output of the encoding is required to be a uniformly random element from $\mathcal{V}^m$. This was shown to also be useful for circuit PSI in [61, 64]. Note that being doubly oblivious directly implies being oblivious as, if the output encoding is an uniformly random element, no adversary may distinguish two different output encodings.

Definition 3 (Doubly Oblivious)**.** *An OKVS is* doubly oblivious *if, for all sets of n distinct keys $\{k_1, \ldots, k_n\} \subseteq \mathcal{K}$ and n values $v_1, \ldots, v_n$ each drawn uniformly at random from $\mathcal{V}$, the encoding* $\mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$ *is statistically indistinguishable from an uniformly random element in $\mathcal{V}^m$.*

Finally, the last property is *random decodings* where the decoded value for a non-input key must be indistinguishable from an uniform random element from $\mathcal{V}$. This property was shown to be useful in building PSU protocols [70].

Definition 4 (Random Decodings)**.** *An OKVS satisfies random decodings if, for all sets of n distinct keys $A = \{k_1, \ldots, k_n\} \subseteq \mathcal{K}$, n values $v_1, \ldots, v_n$ each drawn uniformly at random from $\mathcal{V}$, the output of* $\mathsf{Decode}(\mathbf{S}, k)$ *for key $k \notin A$ is statistically indistinguishable from an uniformly random element in $\mathcal{V}$ where $\mathbf{S} \leftarrow \mathsf{Encode}(\{(k_1, v_1), \ldots, (k_n, v_n)\})$.*

3 OKVS Construction

In this section, we present our construction of random band oblivious key-value stores (RB-OKVS) that are inspired by the family of random band matrices [29]. Additionally, we show a general connection between OKVS and families of random matrices that satisfy certain properties.

3.1 Construction

We start by presenting our construction of RB-OKVS that is parameterized by the number of input keys n, the encoding size $m = (1+\varepsilon)n$ for some small constant $\varepsilon > 0$ and a width parameter w. Furthermore, we will assume that the value universe is a field, $\mathcal{V} = \mathbb{F}$.

High-Level Overview. During the encoding process, RB-OKVS will receive an input of n key-value pairs with n distinct keys, $I = \{(k_1, v_1), \ldots, (k_n, v_n)\} \in (\mathcal{K} \times \mathbb{F})^n$. At a high level, the goal is to construct a matrix $\mathbf{M} \in \{0,1\}^{n \times m}$ using the set of input keys $\{k_1, \ldots, k_n\}$. The i-th row of $\mathbf{M}$, denoted by $\mathbf{M}[i] \in \{0,1\}^m$, will be generated using the i-th input key, k_i. More formally, we will use a hash function $r : \mathcal{K} \to \{0,1\}^m$ such that the matrix $\mathbf{M}$ is defined as

$$\mathbf{M} = \begin{bmatrix} r(k_1)^\intercal \\ r(k_2)^\intercal \\ \cdots \\ r(k_n)^\intercal \end{bmatrix}.$$

We will describe how each row of $\mathbf{M}$ is generated later. Before that, we show how to construct the encoding $\mathbf{s}$ using $\mathbf{M}$ and the input set of key-value pairs I and nice properties we would like to obtain when defining r. Going forward, we will interpret the encoding as a length m vector of elements, $\mathbb{F}^m$.

The goal of the encoding algorithm is to find $\mathbf{s}$ satisfying

$$\mathbf{M} \cdot \mathbf{s} = [v_1, v_2, \ldots, v_n]^\intercal$$

by solving the system of linear equations. For the decoding algorithm, consider any key $k \in \mathcal{K}$ and the encoding $\mathbf{s} \in \mathbb{F}^m$ that was produced by the above encoding. The decoding algorithm first computes the row vector associated to k, $r(k) \in \{0,1\}^m$. Afterwards, the decoding algorithm computes and returns the dot product $r(k) \cdot \mathbf{s}$. By our choice of encoding such that $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$, we know that if $k = k_i$, then $r(k)$ is the i-th row of $\mathbf{M}$, $\mathbf{M}[i]$. As a result,

$$r(k) \cdot \mathbf{s} = r(k_i) \cdot \mathbf{s} = \mathbf{M}[i] \cdot \mathbf{s} = v_i$$

meaning the decoding algorithm correctly returns v_i.

We chose the above structure for our OKVS construction as it is essentially required to satisfy the linearity property needed in PSI applications. For linearity, we see that the decoding algorithm is the dot product of a vector pseudorandomly generated from the query key $r(k)$ and the encoding $\mathbf{s}$. In fact, we will later show that all linear OKVS schemes must satisfy the same structure as RB-OKVS (see Section 3.2).

Choosing matrix M. Before describing our construction, we first describe the desirable properties of the matrix $\mathbf{M}$. First, we need that $\mathbf{M}$ has full row rank with high probability. Otherwise, it is impossible to find an encoding $\mathbf{s}$ satisfying $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$. Additionally, we want that solving for $\mathbf{s}$ such that $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$ is efficient. This is important as generic algorithms for solving systems of equations require at least $O(n^2)$, and typically $O(n^3)$, time. In other words, we want to construct random matrices $\mathbf{M}$ that are efficiently solvable except with very small probability.

To satisfy these requirements, we will construct our matrix $\mathbf{M} \in \{0,1\}^{n \times m}$ using the random band matrix constructions of Dietzfelbinger and Walzer [29]. Random band matrices are generated by ensuring that each row consists of a short random band of width w. All entries outside of the short band will be zero. In more detail, each row is generated by, first, choosing a random entry for the start of the band. Afterwards, a uniformly random w-bit string from $\{0,1\}^w$ is chosen and embedded at the chosen starting entry. The remaining $m - w$ entries are all set to be 0.

Random band matrices are special because they are equipped with a simple and efficient algorithm that enables solving the system in $O(nw + n \log n)$ time. First, the system solving algorithm sorts the rows of $\mathbf{M}$ by the location of the first non-zero entry in the row. Next, one can employ Gaussian elimination with back substitution on the sorted matrix. The key insight is that, during Gaussian elimination, the back substitution only needs to consider a small subset of columns for each row that will be in or nearby the w-bit random band. Furthermore, any columns that do not appear in any row's random band will essentially be skipped as every entry in the column will be zero. As a result, solving the linear system can be done extremely efficiently as we will show later.

With our choice of matrix $\mathbf{M}$, we are now ready to formally present the encoding and decoding algorithms for RB-OKVS.

Algorithm 1 RB-OKVS.Encode algorithm

Input: $I = \{(k_1, v_1), \ldots, (k_n, v_n)\}, R$: n key-value pairs with n distinct keys and randomness.
Output: $\mathbf{s}$: encoding of I.

```
Parse R = (R_1, R_2)        ▷ R_1, R_2 ∈ {0,1}^λ are hash keys.
Initialize M as mapping from indices to non-zero entries.
for i = 1, ..., n do
    a_i ← H_1(R_1, k_i)
    u_i ← H_2(R_2, k_i)
    for j = 1, ..., w do
        M[i][a_i + j − 1] ← u_i[j]
    end for
end for
Sort rows of M according to first non-zero location.
Execute Gaussian elimination with back substitution to
solve for s such that M · s = [v_1, ..., v_n]^T (free variables are
sampled at random for double obliviousness).
if s cannot be computed then
    return s ←$ F^m           ▷ Sample random encoding
end if
return s
```

Encoding. The encoding algorithm of RB-OKVS is formally presented in Algorithm 1 that utilizes the structure of random band matrices described above. The encoding algorithm receives an input set of n key-value pairs with n distinct keys, $I = \{(k_1, v_1), \ldots, (k_n, v_n)\}$ and randomness R. Additionally, recall that RB-OKVS is parameterized by the encoding size $m = (1 + \varepsilon)n$ and width parameter w to generate matrix $\mathbf{M}$.

First, the algorithm will use two random keys R_1 and R_2 from $\{0,1\}^\lambda$ stored in the input randomness R. These will be used for two random hash functions: $\mathsf{H}_1 : \{0,1\}^\lambda \times \mathcal{K} \to \{1, 2, \ldots, m - w\}$ and $\mathsf{H}_2 : \{0,1\}^\lambda \times \mathcal{K} \to \{0,1\}^w$. H_1 will be used to generate the starting location of the band and H_2 will be used to generate the contents of the w-bit band. If H_1 and H_2 are random oracles, we can forego the keys R_1 and R_2.

Next, we construct the function $r : \mathcal{K} \to \{0,1\}^m$ as follows. To compute $r(k)$, we first compute $\mathsf{H}_1(R_1, k) \in \{1, 2, \ldots, m - w\}$ for the starting location and $\mathsf{H}_2(R_2, k) \in \{0,1\}^w$ for the contents of the random band. Then, we set $r(k)[\mathsf{H}_1(R_1, k) + i - 1] = \mathsf{H}_2(R_2, k)[i]$ for all $i \in [w]$ and set $r(k)[j] = 0$ for all $1 \leq j < \mathsf{H}_1(R_1, k)$ and for all $\mathsf{H}_1(R_1, k) + w < j \leq m$. In other words, $r(k)$ is a row with a random w-bit band embedded in a random location.

For any n key-value pairs $I = \{(k_1, v_1), \ldots, (k_n, v_n)\}$, we encode I in the following way. First, we compute $\mathbf{M}$ by setting the i-th row as $\mathbf{M}[i] = r(k_i)$ for all $i \in [n]$. Next, we solve for $\mathbf{s}$ satisfying $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$ by, first, sorting the rows by the starting location of the first non-zero entry in the row and, then, performing Gaussian elimination with back substitution. If the linear system cannot be solved, then the encoding algorithm returns an uniformly random element from $\mathbb{F}^m$. Otherwise, the encoding algorithm returns $\mathbf{s}$.

Algorithm 2 RB-OKVS.Decode algorithm

Input: $\mathbf{s}, k, R$: encoding, key and randomness.
Output: v: value associated with k.

Parse $R = (R_1, R_2)$ $\triangleright$ $R_1, R_2 \in \{0,1\}^\lambda$ are hash keys.
$a \leftarrow \mathsf{H}_1(R_1, k)$
$\mathbf{u} \leftarrow \mathsf{H}_2(R_2, k)$
$v \leftarrow 0$
for $j = 1, \ldots, w$ **do**
  $v \leftarrow v + (\mathbf{u}[j] \cdot \mathbf{s}[a + j - 1])$
end for
return v

We note that the encoding algorithm will never store $\mathbf{M}$ explicitly. As only nw entries are non-zero, we can store $\mathbf{M}$ as a map from indices to non-zero entries using $O(nw)$ space.

Decoding. The decoding algorithm of RB-OKVS is formally defined in Algorithm 2. As input, the decode algorithm receives the encoding $\mathbf{s}$, the query key $k \in \mathcal{K}$ and the randomness R. Using $R = (R_1, R_2)$, the decoding algorithm can construct the same random functions H_1 and H_2.

To decode the value associated with query key k, we first compute $\mathsf{H}_1(R_1, k)$ and $\mathsf{H}_2(R_2, k)$. Next, we compute the following dot product between the w-bit vector $\mathsf{H}_2(R_2, k)$ and a consecutive w-bit subsequence of $\mathbf{s}$ as follows:

$$\sum_{i=1}^{w} \mathsf{H}_2(R_2,k)[i] \cdot \mathbf{s}[\mathsf{H}_1(R_1,k) + i - 1].$$

Note, this computation is equivalent to computing $r(k) \cdot \mathbf{s}$ as we know that, for all $1 \le j < \mathsf{H}_1(R_1, k)$ and $\mathsf{H}_1(R_1, k) + w < j \le m$, the j-th entry of $r(k)$ is zero, $r(k)[j] = 0$.

To see correctness, we note that if $k = k_i$, then we know that $r(k) = r(k_i) = \mathbf{M}[i]$. The response of the decoding algorithm is $r(k) \cdot \mathbf{s} = \mathbf{M}[i] \cdot \mathbf{s}$. Assuming that the encoding algorithm returned $\mathbf{s}$ satisfying $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$, the decoding algorithm returns the right answer as $\mathbf{M}[i] \cdot \mathbf{s} = v_i$.

(Doubly) Obliviousness. One important property for circuit PSI is being doubly oblivious where the output encoding must be indistinguishable from an uniformly random element of $\mathbb{F}^m$. We show that RB-OKVS is indeed doubly oblivious. Recall (from Section 2.1) also that doubly obliviousness implies obliviousness.

Recall that while solving for $\mathbf{s}$ satisfying $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$ in the encoding algorithm, the last step is to perform Gaussian elimination with back substitution. In this process, since the number of columns m is greater than the number of rows n, we are left with some *free* variables in $\mathbf{s}$, whose values we can choose arbitrarily. To obtain doubly obliviousness, we choose these values randomly (whereas before, we could set them to, say, 0 for efficiency). Then, starting from row n to row 1, back substitution solves for the *lead* variables s_i of each row j (i.e., the first non-zero entry in each row j) in terms of previously chosen $s_{i'}$, $i' > i$, and some subset $\{v_{j_1}, \ldots, v_{j_k}\} \subseteq \{v_1, \ldots, v_n\}$, containing v_j. Since, v_j is chosen uniformly at random according to the definition of double obliviousness, s_i will be distributed uniformly, regardless of all other values $s_{i'}$, $i' > i$, and $v_{j_l} \neq v_j$. Thus, it is clear that this construction satisfies double obliviousness.

Analysis. We start by analyzing the correctness of RB-OKVS. The decoding algorithm works correctly assuming that the matrix $\mathbf{M}$ is solvable. For $w = O(\log n)$, it was shown that $\mathbf{M}$ is solvable in time $O(nw)$ except with probability $O(1/n)$ (see [29]). In Appendix A, we extend this result to larger bands to show that for any $w = O(\lambda)$, the matrix $\mathbf{M}$ is solvable in time $O(n\lambda)$ except with probability $2^{-\lambda}$. For concrete instantiations of parameters m and w, see Section 6.1.

For the efficiency of RB-OKVS, we start with the encoding. The first step of the encoding algorithm is to sort the rows by the starting index of the random band that can be done in $O(n \log n)$ time using radix sort. Note, the total time becomes $O(n\lambda)$ as $\lambda = \omega(\log n)$ to obtain negligible error. Next, the encoding algorithm solves the system of equations defined by the matrix $\mathbf{M}$ that requires $O(n)$ time for sorting as all column indices are elements in $[m] = [(1+\varepsilon)n]$ and $O(nw)$ time for Gaussian elimination for a total of $O(nw)$ time. Decoding requires time $O(w)$ as it computes a dot product of two w-length vectors.

For the additional properties, we already outlined why RB-OKVS is linear and doubly oblivious. As $\mathbf{M}$ is binary, we note that RB-OKVS is also binary. For random decodings, we note that each element in $\mathbf{s}$ is an uniformly random element from $\mathbb{F}$ since RB-OKVS is doubly oblivious. For any key k, as long as $r(k)$ contains at least one non-zero entry, then the decoding result is a random element in $\mathbb{F}$. We note that $r(k)$ contains a w-bit random binary string that will be all 0 only with probability 2^{-w}.

We show that RB-OKVS satisfies the following:

Theorem 1. *If $w = O(\lambda/\varepsilon + \log n)$ and $m = (1+\varepsilon)n$ for some $\varepsilon > 0$, then* RB-OKVS *is an OKVS with error probability $2^{-\lambda}$, encoding time $O(nw)$ and decoding time $O(w)$. Assuming random oracles,* RB-OKVS *is oblivious and doubly oblivious with random decodings except with probability 2^{-w}. Finally,* RB-OKVS *is both binary and linear.*

The full proof of this theorem may be found in Appendix A.

Word Operations. In practice, we note that w is quite small and can fit into a constant number of words. As a result, we can utilize SIMD operations to perform row additions requiring $O(w)$ time using only $O(1)$ word operations. The encoding and decoding times of RB-OKVS can be viewed as linear, $O(n)$, and constant, $O(1)$, respectively in practical settings. We will utilize SIMD operations for RB-OKVS in our implementations (see Section 6.1).

Cache Efficiency. As identified in [61], an important measure of efficiency for OKVS constructions is the cache efficiency. For large amounts of data where the CPU cache can no longer store the entirety of the data, the OKVS must retrieve missing data from slower main memory.

RB-OKVS was designed to be cache-friendly. Recall that the encoding algorithm requires sorting and performing Gaussian elimination. For sorting, we choose a cache-friendly algorithm. For Gaussian elimination, rows are processed in sequential order (without random access) meaning that data can be fetched with the minimal number of main memory lookups. For decoding, only a consecutive subsequence of w elements are required for decoding any key. These are key reasons that the computational cost of RB-OKVS is small. See Section 6.1 for experimental evaluation.

Binary vs. Non-Binary. In our construction, we use random band matrices where entries are either 0 or 1. A natural extension would be to consider bands where each entry is a random element from a field $\mathbb{F}$. In practice, binary random band matrices are more efficient as solving the linear system will mainly use bit-wise operations (such as XOR) whereas band matrices with random field elements will need to use field operations that are slower in practice.

3.2 Connection with Random Matrices

In this section, we show that one can generalize our above construction for general families of random matrices. Additionally, we show that the structure of RB-OKVS that solves a linear system of equations for some binary matrix $\mathbf{M}$ is required for any linear OKVS like RB-OKVS.

General Framework. We show that we can generalize the construction of RB-OKVS by replacing random band matrices with random binary matrix families satisfying special properties. Let $\mathcal{F} = \{\mathbf{M}_1, \ldots, \mathbf{M}_\ell\} \subseteq \{0,1\}^{n\times m}$ be a matrix family that is equipped with an algorithm $\mathcal{A}_\mathcal{F}$ that, given $\mathbf{M}_i$ and $\mathbf{v} \in \mathbb{F}^n$, outputs $\mathcal{A}_\mathcal{F}(\mathbf{M}_i, \mathbf{v}) = \mathbf{s}$ such that $\mathbf{M}_i \cdot \mathbf{s} = \mathbf{v}$. Additionally, suppose there is a random mapping $r_\mathcal{F} : \mathcal{K} \to \{0,1\}^m$ from keys to row vectors. Furthermore, for any set of n keys $\{k_1, \ldots, k_n\}$, the following matrix $\mathbf{M}$ is a member of $\mathcal{F}$:

$$\mathbf{M} = \begin{bmatrix} r_\mathcal{F}(k_1)^\intercal \\ r_\mathcal{F}(k_2)^\intercal \\ \ldots \\ r_\mathcal{F}(k_n)^\intercal \end{bmatrix}.$$

We can construct $\mathcal{F}$-OKVS by essentially replacing the row generating function r and algorithm for solving random band matrices in RB-OKVS with the above mapping $r_\mathcal{F}$ and algorithm $\mathcal{A}_\mathcal{F}$. Any improved constructions of these matrix families would immediately imply better OKVS schemes.

Known Matrix Families. Beyond random band matrices, we note that there are other known matrix families satisfying the requirements. Garimella *et al.* [34] used the framework above to construct an OKVS with the family of uniformly random binary matrices. While random binary matrices only require $m = n + O(\log n)$ to ensure solvability, the best known solving algorithm requires $O(n^3)$ time in practice. Raghuraman and Rindal [61] constructed another such matrix family for their OKVS scheme. To our knowledge, random band matrices [29] remain the most efficient to date.

Optimality of Approach. An interesting question is whether the above framework of using these random binary matrix families is optimal. For example, is it possible to construct better OKVS using a different encoding algorithm? For the case of linear OKVS, we show that there is no such better approach and that finding such matrix families is equivalent to building a linear OKVS.

To see why, we can analyze the properties of an OKVS being linear. For an input $I = \{(k_1, v_1), \ldots, (k_n, v_n)\}$, suppose a binary and linear OKVS outputs encoding $\mathbf{s}$. Then, there exists some function $d : \mathcal{K} \to \mathbb{F}^m$ such that $d(k_i) \cdot \mathbf{s} = v_i$ for all $i \in [n]$. Consider the matrix

$$\mathbf{M} = \begin{bmatrix} d(k_1)^\intercal \\ d(k_2)^\intercal \\ \ldots \\ d(k_n)^\intercal \end{bmatrix} \in \mathbb{F}^{n\times m}$$

and note that it satisfies $\mathbf{M} \cdot \mathbf{s} = [v_1, \ldots, v_n]^\intercal$. Furthermore, the Encode algorithm immediately provides an algorithm for solving linear systems associated with these matrices. In other words, any linear OKVS immediately emits a family of random matrices that are efficiently solvable (see full version for more details). If we consider OKVS schemes that are both linear and binary, then the same arguments holds for families of random binary matrices.

4 OKVS for Private Set Operations

In this section, we outline important applications of the OKVS primitive beyond volume-hiding multi-maps. In particular, we will describe prior works that have used the OKVS primitive for various private set operations.

Private Set Intersection (PSI). Prior works [34,61] identified that OKVS are integral for building efficient PSI protocols. In [34], it was shown that PSI with both semi-honest and malicious security may be built using any linear OKVS. A technique from [64] can further improve this maliciously secure protocol to have essentially no overhead compared to its semi-honest variant. Finally, it was shown that circuit PSI protocols can be built from any doubly oblivious OKVS [61].

By plugging in RB-OKVS as the underlying OKVS to the above frameworks, we obtain our new semi-honest, malicious and circuit PSI protocols reducing total communication and monetary cost (see Section 6.2). Formal functionalities of cryptographic primitives related to PSI and descriptions of the PSI constructions using an OKVS may be found in the full version.

Private Set Union (PSU). OKVS have also been shown to be important for building PSU protocols [51,70]. In particular,

	Server Storage	Request	Response
DST [46]	$O(n)$	$O(1)$	$O(\ell \log n)$
dprfMM [54]	$(2+\varepsilon)n$	$O(1)$	2ℓ
$\mathcal{S}^4$ [68]	$2n$	$O(1)$	ℓ
XorMM [69]	$1.23n$	$O(1)$	ℓ
Ours: RB-MM	**1.03n**	$O(1)$	ℓ

Figure 2: Comparison of lossless VH-EMM constructions.

the state-of-the-art PSU protocol is built using OKVS with the random decodings property [70]. Using RB-OKVS, we obtain our new PSU protocol with improved communication, computation and monetary cost (see Section 6.3). A formal description of the PSU functionality and construction from OKVS may be found in the full version.

Other Applications. We note that the above applications of intersection and union are two of the simplest private set operations where OKVS are integral. We also expect that the OKVS primitive will be important for more complex set operations that we leave to future work.

5 Volume-Hiding Encrypted Multi-Maps

We show that one can build a volume-hiding encrypted multi-map (VH-EMM) using any OKVS. To our knowledge, this is the first connection between VH-EMM and OKVS. Plugging RB-OKVS into our framework, we also obtain state-of-the-art volume-hiding encrypted multi-maps (see Figure 2).

Encrypted Multi-Maps (EMM). To start, we define encrypted multi-maps (EMM) that is a form of structured encryption [21] where the goal is to outsource data to an untrusted server while still being able to query the data. For privacy, all information about the data should be hidden except some well-defined and sensible leakage function.

For an EMM, the data is represented as a multi-map consisting of pairs of keys and value tuples, $I = \{(k_1, \mathbf{v}_1), \ldots, (k_n, \mathbf{v}_n)\} \in \{\mathcal{K} \times \mathcal{V}^*\}^n$. Each key may be associated to multiple values unlike in an OKVS. An EMM should also support querying the value tuple for any key $k \in \mathcal{K}$.

The study of EMMs is important due to its use in several applications. An EMM may be used as an encrypted index for searchable encryption [66] over corpora of encrypted documents and SQL queries over encrypted databaes [45].

Volume-Hiding EMMs (VH-EMM). An important line of work is to understand the implications of leakage functions using attacks for specific leakage profiles [42]. To protect against these attacks, the notion of a volume-hiding EMM, VH-EMM, was introduced [46] where the goal is to guarantee that the number of values (volume) associated with any key is hidden from the server. VH-EMM protect against any abuse attacks that rely on volume leakage. We point readers to Section 7 for more discussion on related works.

Algorithm 3 RB-MM.Setup algorithm

Input: $I = \{(k_1, \mathbf{v}_1), \ldots, (k_n, \mathbf{v}_n)\}$: input multi-map
Output: $\mathsf{st}, \mathsf{EMM}$: client state and encrypted multi-map
Sample random keys K^F and K^E
$I' \leftarrow []$
for $i = 1, \ldots, n$ **do**
  $h_i \leftarrow \mathsf{F}(K^F, k_i)$
  for $j = 1, \ldots, |\mathbf{v}_i|$ **do**
    $a_{i,j} \leftarrow h_i \,||\, j$
    $b_{i,j} \leftarrow \mathsf{Enc}(K^E, h_i \,||\, \mathbf{v}_i[j])$
    $I' \leftarrow I' \cup \{(a_{i,j}, b_{i,j})\}$
  end for
end for
Sample random keys $R \leftarrow (R_1, R_2)$
$\mathsf{EMM} \leftarrow \mathsf{RB\text{-}OKVS.Encode}(I', R)$
$\mathsf{st} \leftarrow (K^F, K^E)$
return $(\mathsf{st}, (\mathsf{EMM}, R))$

5.1 Construction

We show that we can construct our VH-EMM, RB-MM, using RB-OKVS = (Encode, Decode). Although, one can use any OKVS satisfying random decodings to replace RB-OKVS. The pseudocode of our constructions for the setup and query algorithm are found in Algorithm 3 and 4 respectively.

Setup. Given input $I = \{(k_1, \mathbf{v}_1), \ldots, (k_n, \mathbf{v}_n)\}$ with maximum volume ℓ, the setup algorithm first samples PRF and encryption keys K^F and K^E. Next, all input keys are hashed and all value tuples are encrypted. Finally, the input set is flattened such that each value is keyed by the original key and the value's index in the tuple. For example, the j-th value in the i-th pair, $(k_i, \mathbf{v}_i[j])$ is converted into the following pair:

$$(\mathsf{F}(K^F, k_i) \,||\, j, \mathsf{Enc}(K^E, \mathbf{v}_i[j]))$$

where Enc is an authenticated encryption scheme. Denote the flattened set I'. Then, we encode using the OKVS, $\mathsf{EMM} = \mathsf{RB\text{-}OKVS.Encode}(I', R)$, that is sent to the server where R is randomness generated by the client and also shared with the server.

Query. To query for a key $k \in \mathcal{K}$, the client uploads the PRF evaluation $\mathsf{F}(K^F, k)$ to the server. The server executes the decoding algorithm of the OKVS ℓ times as follows:

$$\{\mathsf{RB\text{-}OKVS.Decode}(\mathsf{EMM}, \mathsf{F}(K^F, k) \,||\, i, R) \mid i \in [\ell]\}$$

and returns the ℓ responses to the client. The client decrypts the ℓ responses and ignores all failed decryptions.

Discussion about Prior Works. In the above, we showed that one can build an VH-EMM using any OKVS. To our knowledge, this is the first connection between the two primitives.

Algorithm 4 RB-MM.Query algorithm

Input: $k, \mathsf{st}, \mathsf{EMM}, R$: query key, client state, encrypted multi-map and randomness

Output: $\mathbf{v}$: value tuple

```
Parse st = (K^F, K^E)                          ▷ Executed by client
h ← F(K^F, k)
Send h to server

X ← []                                         ▷ Executed by server
for i = 1, ..., ℓ do
    x_i ← RB-OKVS.Decode(EMM, h || i, R)
    X ← X ∪ {x_i}
end for
Send X to client

Parse X = (x_1, ..., x_ℓ)                      ▷ Executed by client
v ← []
for i = 1, ..., ℓ do
    y_i ← Dec(K^E, x_i)
    If y_i does not start with h, terminate loop.
    v ← v ∪ {y_i}
end for
return v
```

However, we note that prior works have implicitly used similar ideas with close variants of known OKVS. For example, XorMM [69] utilized a very close variant of 3H-GCT [34] from cuckoo hashing with 3 hash functions. In this work, we make this connection between VH-EMM and OKVS explicit.

5.2 Analysis

Efficiency. Consider an instantiation of RB-OKVS by setting $w = O(\lambda)$ to obtain $2^{-\lambda}$ error probability. First, we note that the resulting EMM has size $1.03n$ that is a 16% improvement over $1.23n$ of the best prior work [69]. For query communication, only a single PRF evaluation is uploaded and exactly ℓ values are downloaded. The query time is $O(\ell\lambda)$ as we perform ℓ decode operations. See Figure 2 for comparisons.

Correctness. For correctness, we only need to rely on the random decodings property of RB-OKVS. For any non-input key, we note that the decryption of the authenticated encryption scheme succeeds only with negligible probability.

Security and Volume-Hiding. For volume-hiding, we note that RB-OKVS guarantees that the server cannot learn the input keys for random values. As all values are encryptions, we know they are computationally indistinguishable from random for the server. As a result, the server cannot learn the input keys and, thus, the volumes associated with any key $k \in \mathcal{K}$. We formalize the security using the following leakage function $\mathcal{L}(I, Q)$ for any input I and query sequence $Q \in \mathcal{K}^*$. $\mathcal{L}(I)$ consists of the following:

1. Multi-map size: $|I| = |\mathbf{v}_1| + |\mathbf{v}_2| + \ldots + |\mathbf{v}_n|$
2. Maximum volume: ℓ
3. Key-equality matrix: $M \in \{0,1\}^{|Q|\times|Q|}$ such that $M[i][j] = 1$ if and only if $Q[i] = Q[j]$.

The first two are the outputs of $\mathcal{L}_{\mathsf{Setup}}$ while the last is output of $\mathcal{L}_{\mathsf{Query}}$. We note this is the identical leakage as the VH-EMM in prior works including [46, 54, 68, 69].

We show that RB-MM is adaptively secure for this leakage and that this leakage function is volume-hiding as defined in [54]. The formal proof can be found in the full version.

Theorem 2. *Assuming random oracles,* RB-MM *is adaptively secure with respect to $\mathcal{L}$ and $\mathcal{L}$ is volume-hiding.*

6 Experimental Evaluation

In this section, we perform experiments to evaluate RB-OKVS as well as the application of RB-OKVS to multiple cryptographic primitives including private set intersection (PSI), private set union (PSU) and volume-hiding encrypted multi-maps (VH-EMM).

Experimental Setup. We conducted our experiments using Ubuntu PCs with 12 cores, 3.7 GHz Intel Xeon W-2135 and 64 GB of RAM. We use the AVX2 and AVX-512 instruction sets with SIMD instructions enabled and only single-threaded execution. All reported results are the average of at least 10 experimental trials with standard deviation less than 10% of the averages. Monetary costs are computed using Amazon EC2 pricing of t2.2xlarge instances [2] of \$0.09 per GB of outbound traffic and \$0.014 per CPU hour at the time.

Security Level and Error Probability. In our constructions and experiments, we will choose parameters appropriately to ensure 40 bits of statistical security and 128 bits of computational security (following [61]). Parameters will be chosen such that the OKVS error probability is at most 2^{-40}.

6.1 Oblivious Key-Value Stores

First, we perform experiments to pick concrete parameters for RB-OKVS to obtain the desired security and error probabilities. Afterwards, we compare with prior OKVS schemes.

Concrete Instantiation. We evaluate the concrete failure probability guarantees provided by RB-OKVS. Recall that RB-OKVS consists of two main parameters: the encoding size $m = (1+\varepsilon)n$ and the band length w. For any choices of ε and w, we will aim to determine the statistical security parameter λ such that the failure probability of RB-OKVS is at most $2^{-\lambda}$. Afterwards, we pick parameters to obtain $\lambda = 40$.

To do this, we use an analytical evaluation similar to prior OKVS works (such as [61]). We will consider choices of w for small security parameters λ. In particular, we verify

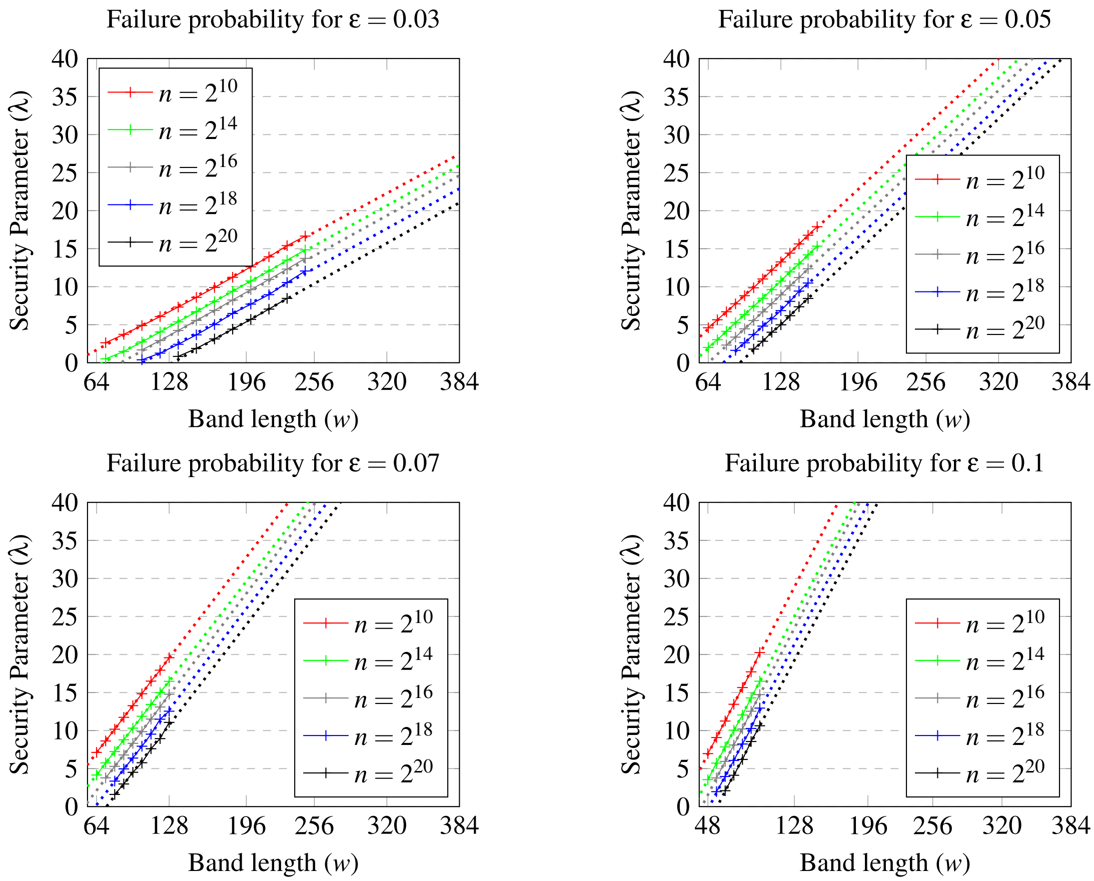

Figure 3: Security parameters based on band lengths with extrapolated lines based on linear regression on sampled data points.

our parameters by seeing that the slope of security parameter to band length (λ/w) is consistent for smaller security parameters giving us confidence that the slope will remain the same for larger security parameters that we cannot verify. In Figure 3, we present the statistical security parameter λ as a function of the band length w for various choices of $\varepsilon \in \{0.03, 0.05, 0.07, 0.1\}$ resulting in rates of 0.91-0.97. We also plot dotted lines representing extrapolated lines for larger security parameters. For a fixed ε, the growth of λ with respect to ε seems to be the same for any n suggesting that the slope depends only on ε. This corroborates with our theoretical analysis in Theorem 1 stating that $w = O(\lambda/\varepsilon + \log n)$ is required to obtain λ statistical security. In other words, we can re-arrange this as $\lambda = O(\varepsilon \cdot w - \log n)$ and the slope (λ/w) depends only on ε. We plot the slopes (λ/w) for various ε and n in Figure 4. It can be observed that the plots for various choices of $n \in \{2^{10}, 2^{14}, 2^{16}, 2^{18}, 2^{20}\}$ are essentially the same further exhibiting that the slope depends exclusively on ε. In particular, we can estimate the slope as being 2.751ε, and get the following formula to pick w for specific λ:

$$\lambda = 2.751\varepsilon w + g(\varepsilon, n):$$

Above, g is a function for the y-intercepts. To get an idea on how g behaves, we plot the y-intercepts of the best fit lines in Figure 5. Looking at this plot, it's not clear how we can approximate it as a general function that will give us a pessimistic lower bound on λ. Instead, we take the following approach: for fixed values of ε and n, we run the failure probability experiment to obtain (small) λ for an arbitrary choice of the band width w. Then, we plug in these values to the equation $\lambda = 2.751\varepsilon w + g(\varepsilon, n)$ to obtain the value of $g(\varepsilon, n)$, and use this as our choice of the y-intercept. We point out that for $n = 2^{24}$ in our experimental evaluation, we have precisely used this procedure to compute the band width that gives us $\lambda = 40$ bits of security.

We provide the best fit lines from our experiments in Appendix B for reference and future usage.

Finally, we revisit the case of trying to choose both ε and w given fixed choices of input size n and target statistical security λ. In general, ε is the main parameter that enables trade-offs between the rate and encoding/decoding times of RB-OKVS. For small encoding sizes (i.e., high rate), one should try to fix small choices of ε such as 0.03-0.05 and, then, run the strategy above to pick a sufficient w for the target security level λ. In contrast, if one wishes for an instantiation of RB-OKVS with fast encoding/decoding times, then one can pick larger values of ε such as 0.07-0.1. We use this strategy to obtain various protocols that perform better in different

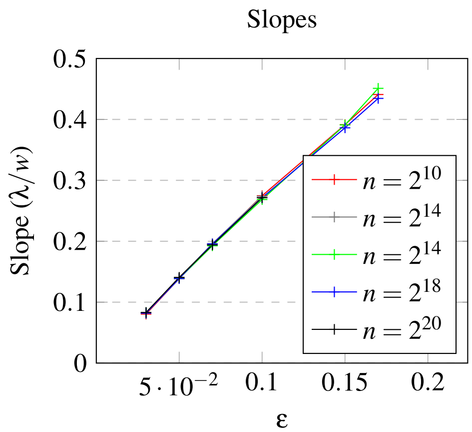

Figure 4: Slopes of best fit lines of the failure probability.

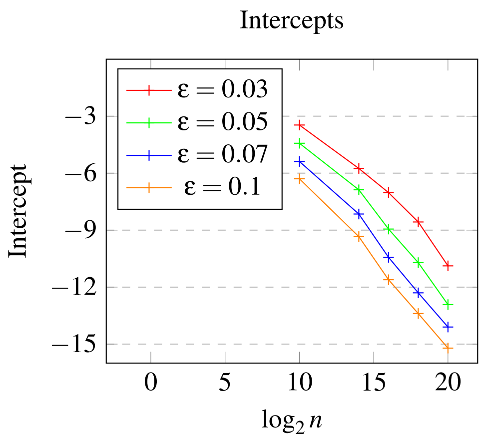

Figure 5: Intercepts of best fit lines of the failure probability.

network settings for PSI in Section 6.2.

Comparison. Next, we compare RB-OKVS with prior constructions of OKVS: 3H-GCT [34] and RR22 [61]. To evaluate our OKVS construction RB-OKVS, we consider four different choices of $\varepsilon \in \{0.03, 0.05, 0.1, 0.15\}$. For 3H-GCT [34], we evaluate both their standalone construction as well as one that amplifies security using their star architecture. For RR22 [61], we also evaluate two constructions with different encoding sizes (rates) of $1.28n$ (0.78) and $1.23n$ (0.81). We denote these two constructions as RR22 (fast) and RR22 (small) respectively. To our knowledge, RR22 (small) [61] is the best rate achievable by prior works with linear encoding times. All constructions target 40 bits of statistical security. RB-OKVS and RR22 [61] consider 128-bit elements while 3H-GCT [34] considers only 64-bit elements. All results are presented in Figure 6 including the encoding size (rate) as well as the encoding, batch decoding and total times. By batch decoding, we mean that all n input keys are decoded.

First, we see that the encoding size and rate of all four instantiations of RB-OKVS are better than all instantiations of 3H-GCT [34] and RR22 [61]. The smallest instantiation of RB-OKVS with $\varepsilon = 0.03$ achieves near-optimal rate of 0.97. Even with this significant size improvement, the total running time of this instantiation is 17-23x faster than 3H-GCT. Compared to RR22 (small), the total time of RB-OKVS ($\varepsilon = 0.03$) is slightly faster than RR22 (small), but slower than RR22 (fast). In our opinion, this still seems like a reasonable trade-off given that this instantiation of RB-OKVS ($\varepsilon = 0.03$) has 20% smaller encoding size than RR22 (fast).

Next, we also show the flexibility of RB-OKVS by presenting instantiations with larger rate but faster running times. Instantiating RB-OKVS with larger ε will result in smaller running times. This trade-off between rate and running time is achieved very efficiently by RB-OKVS. For any desired rate of at most 0.97, RB-OKVS runs faster than any prior OKVS construction. This can be seen by the fact that all four instantiations of RB-OKVS in Figure 6 have smaller total time than the construction achieving the best rate of prior works, RR22 (small), even though all four RB-OKVS instantiations achieve better rate than RR22 (small).

One drawback of RB-OKVS is that the decoding time is larger than RR22 [61]. This is inevitable as decoding of RB-OKVS involves more entries compared to RR22 [61]. If one wishes for very fast decoding, then RR22 remains better than RB-OKVS. However, we note that the encoding times of RB-OKVS are significantly faster. The total time of all RB-OKVS instantiations remains faster than RR22 (small).

Discussion about Small ε. In our experiments, we considered ε as small as 0.03 that results in rate 0.97. One may wonder what happens if we considered very small ε approaching 0 towards optimal rates of 1. As evidenced by above, the band length w will continue to increase rapidly as ε decreases. The encoding and decoding times are directly related to the band length w. Therefore, smaller ε will result in less efficient encoding and decoding algorithms. In the extreme case, if ε becomes so small, the band length w will become as large as each row. For this setting, the resulting matrix is essentially a uniformly random binary matrix that requires cubic time to solve the linear system in practice.

6.2 Private Set Intersection

In this section, we present experimental evaluations for our PSI protocols and compare with prior works. We evaluate all the constructions across three network settings: fast networks with 1 Gib/sec, a medium network with 260 Mib/sec and a slow network with 33 Mib/sec (following [34, 61]).

The results of our experimental evaluations may be found in Figure 7. All our PSI constructions plug our implementation of RB-OKVS into the PSI implementation of [61] found at [6]. For more details formal descriptions, we point readers to Section 4 and the full version. For prior constructions except RR22 [61], we use reported results from prior works (fast network results from [61] and medium/slow network results from [34]). For [61], we execute the implementation found at [6]. All reported results for our PSI protocols from

Construction	Encoding Size (Rate)	Encode (ms)			Batch Decode (ms)			Total (ms)		
		2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}
3H-GCT [34]	$1.3n$ (0.77)	383	9,009	166,734	203	3,573	63,547	586	12,582	230,281
3H-GCT [34] (star)	$1.32n$ (0.76)	460	5,388	-	309	5,532	-	769	10,920	-
RR22 [61] (fast)	$1.28n$ (0.78)	20	188	2,710	3	49	989	23	237	3,699
RR22 [61] (small)	$1.23n$ (0.81)	24	540	9,874	**2**	**50**	**782**	26	590	10,464
Ours: RB-OKVS ($\varepsilon = 0.03$)	**1.03n (0.97)**	24	365	6,698	10	168	3,170	34	533	9,868
Ours: RB-OKVS ($\varepsilon = 0.05$)	$1.05n$ (0.95)	14	232	3,911	7	115	2,311	21	347	6,222
Ours: RB-OKVS ($\varepsilon = 0.1$)	$1.10n$ (0.91)	12	148	2,159	6	109	1,658	18	257	3,817
Ours: RB-OKVS ($\varepsilon = 0.15$)	$1.15n$ (0.87)	**12**	**133**	**1,958**	5	103	1,557	**17**	**236**	**3,515**

Figure 6: Comparison of RB-OKVS and prior OKVS constructions with 40 bits of statistical security. Elements are 128 bits for our constructions and RR22 [61] and 64 bits for 3H-GCT [34]. Total is the sum of the encode and batch decode times. Bolded numbers mark the best values.

RB-OKVS and RR22 [61] were executed in our experimental setup described in Section 6.1. All our results are presented in Figure 7. In Figure 8, we present the monetary costs for PSI protocols in the fast network setting.

Semi-Honest PSI from RB-OKVS. As can be seen in Figure 7, the communication costs of our PSI protocols using RB-OKVS are 8-12% smaller than RR22 [61]. This is unsurprising given that RB-OKVS has higher rate than prior OKVS schemes. In slow network settings (33 Mb/s), our PSI protocols are up to 8% faster than prior works due to the smaller communication. For faster networks, our PSI protocols are competitive but slower than RR22 [61]. We attribute the increase computation due to multiple decodings of the OKVS as RB-OKVS has slower decoding than the OKVS used in [61]. However, our PSI protocols are up to 13% more cost-efficient than RR22 [61] even in fast networks due to the smaller communication costs (see Figure 8).

Malicious PSI from RB-OKVS. The same improvements in communication and costs can also be seen in the malicious PSI setting. Our malicious PSI protocol with RB-OKVS use 10% less communication than the malicious protocol in [61]. Furthermore, our PSI protocols are 6% faster in slow network settings. For faster networks, our PSI protocols are slower than RR22 [61]. Our malicious PSI protocols with RB-OKVS are 11% more cost-efficient than RR22 [61] even with fast networks (1 Gib/s).

Circuit PSI from RB-OKVS. Finally, we also obtain similar improvements for circuit PSI using RB-OKVS as the underlying OKVS. Our circuit PSI with RB-OKVS has 12% smaller communication compared to RR22 [61]. Similarly, our protocol is faster in slow network settings, but slower in medium and fast networks compared to [61] that is caused by the slower decoding of RB-OKVS. Overall, our circuit PSI reduces monetary cost by 9% due to the smaller communication despite the increase in computation.

6.3 Private Set Union

Next, we present experimental comparisons of our PSU protocols with RB-OKVS and prior instantiations. We will compare with the PSU protocols in [33, 51, 70] and use the implementations found at [4]. Recall that our PSU protocols are obtained by plugging RB-OKVS into the prior PSU framework [70] (see Section 4 and the full version).

For our PSU implementations, we plug in our construction of RB-OKVS into the prior PSU implementation found at [4] and compare with prior instantiations also found at [4]. For [70], we consider their three instantiations SKE-PSU, PKE-PSU and PKE-PSU* using secret-key and public-key operations respectively. The difference between PKE-PSU and PKE-PSU* is that PKE-PSU* will not perform point compression. We configure RB-OKVS with $\varepsilon = 0.03$ and 40 bits of statistical security and denote our new PSU protocols as RB-SKE-PSU, RB-PKE-PSU and RB-PKE-PSU*.

All our reported results are presented in Figure 9 in the network setting of 1 Gbits/sec. We note that the variants using RB-OKVS has both smaller online communication, less online time and lower costs compared to their counterparts. In particular, RB-PKE-PSU and RB-PKE-PSU* obtain 16-22% improvements in online communication and 28-40% faster online times compared to PKE-PSU and PKE-PSU* respectively. RB-SKE-PSU obtains more modest improvements of 2% in online communication and 6-9% in online time compared to SKE-PSU. These improvements can be attributed to the fact that [70] utilized a variant of 3H-GCT [34] as their OKVS. As seen in Figure 6, RB-OKVS significantly outperforms 3H-GCT in both rate and running time resulting in our improved PSU protocols. We also expect our RB-OKVS variants of PSU to perform better than prior constructions in more network constrained settings due to the smaller communication costs. The offline communication and offline times remain the same as the OKVS schemes are only used in the online phases of the PSU protocols.

Comparing to the PSU protocols in [33, 43, 51], our RB-OKVS variants obtain smaller communication and faster

Construction	Comm. (MB)			1 Gbits/sec			260 Mbits/sec			33 Mbits/sec		
	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}
Semi-Honest PSI												
KKRT16 [49]	8.06	132.12	2,164.26	137	2,073	53,933	2,309	12,568	-	10,220	146,067	-
PaXoS [56]	9.90	166.20	2,703.05	763	4,998	123,800	1,395	11,935	-	8,448	60,159	-
RS21 [64]	7.49	55.84	834.67	499	4,580	113,994	-	-	-	-	-	-
3H-GCT [34] (star)	6.39	103.28	1,686.11	180	2,268	-	1,343	9,504	-	9,491	34,870	-
RR22 [61]	2.25	32.44	530.96	123	**950**	**14,046**	**173**	**1,716**	**26,227**	646	8,741	142,483
RB-OKVS ($\varepsilon = 0.05$)	**2.05**	**28.48**	**466.80**	138	1,955	20,284	197	1,962	31,096	**624**	**8,153**	**131,964**
RB-OKVS ($\varepsilon = 0.1$)	2.10	29.32	480.22	**108**	1,162	16,599	185	1,857	27,481	640	8,246	132,228
Malicious PSI												
PaXoS [56]	14.47	231.47	3,703.57	769	5,196	126,294	2,119	12,042	-	8,152	60,771	-
RS21 [64]	7.86	62.19	918.55	556	5,228	132,951	-	-	-	-	-	-
3H-GCT [34] (star)	11.12	177.86	2,845.83	184	2,291	-	1,343	9,504	-	9,491	34,870	-
RR22 [61]	2.71	38.73	614.85	**112**	**1,044**	**14,959**	**190**	**1,886**	**29,056**	763	10,364	162,776
RB-OKVS ($\varepsilon = 0.05$)	**2.51**	**34.77**	**550.69**	121	2,131	21,087	207	2,140	33,751	**746**	**9,657**	153,070
RB-OKVS ($\varepsilon = 0.1$)	2.56	35.61	564.11	131	1,247	17,512	208	2,026	30,419	750	9,811	**152,190**
Circuit PSI												
RS21 [64] (IKNP)	179.31	1,918.89	-	1,810	25,300	-	-	-	-	-	-	-
RS21 [64] (SilentOT)	22.13	290.46	-	5,021	112,421	-	-	-	-	-	-	-
CGS22 [20] (IKNP+)	68.58	1,160.77	-	2,851	28,723	-	-	-	-	-	-	-
RR22 [61] (Silver)	7.56	120.65	-	**1,304**	**17,689**	-	**1,350**	**18,002**	-	1,684	22,475	-
RB-OKVS ($\varepsilon = 0.05$)	**6.66**	**105.86**	-	1,320	19,387	-	1,378	18,514	-	**1,667**	**21,978**	-
RB-OKVS ($\varepsilon = 0.1$)	6.81	108.22	-	1,319	17,999	-	1,365	18,217	-	1,675	22,025	-

Figure 7: Comparison of PSI protocols with all times reported in milliseconds. Numbers for all protocols except RR22 and our PSI protocols using RB-OKVS are from prior works (1 Gbits/sec from [61] and 260 Mbits/sec and 33 Mbits/sec from [34]). Our circuit PSI schemes using RB-OKVS are built using Silver as the underlying OT. Bolded numbers mark the best values.

Construction	Cost ($)		
	2^{16}	2^{20}	2^{24}
Semi-Honest PSI			
RR22 [61]	< 0.001	0.003	0.048
RB-OKVS ($\varepsilon = 0.05$)	< 0.001	**0.002**	**0.042**
RB-OKVS ($\varepsilon = 0.1$)	< 0.001	**0.002**	0.043
Malicious PSI			
RR22 [61]	< 0.001	0.003	0.055
RB-OKVS ($\varepsilon = 0.05$)	< 0.001	0.003	**0.049**
RB-OKVS ($\varepsilon = 0.1$)	< 0.001	0.003	0.051
Circuit PSI			
RR22 [61] (Silver)	< 0.001	0.011	-
RB-OKVS ($\varepsilon = 0.05$)	< 0.001	**0.010**	-
RB-OKVS ($\varepsilon = 0.1$)	< 0.001	**0.010**	-

Figure 8: Comparison of costs in 1 Gbits/sec network. Bolded numbers mark the best values.

online times. In contrast, the protocols in [33, 43, 51] have more lightweight offline phases. However, both the total communication and running time of our PSU protocols using RB-OKVS remain smaller than [33, 43, 51]. This is expected as the PSU protocols in [70] obtained more efficient online phases by using heavier offline phases.

To our knowledge, RB-PKE-PSU* is the PSU protocol with the smallest online communication and total monetary cost. RB-SKE-PSU results in the PSU protocol with the smallest online time to date. Unfortunately, the offline time of RB-SKE-PSU increases significantly for larger sets sizes. As a result, RB-PKE-PSU* seems to be a better alternative for a PSU protocol with fast online times for larger input sets.

6.4 Volume-Hiding Encrypted Multi-Maps

We implement RB-MM and perform experimental evaluation to compare with the implementations of dprfMM [54] and XorMM [69] in [7] using same experimental setup as specified in Section 6.1. To instantiate the underlying RB-OKVS, we choose $\varepsilon = 0.03$ and the necessary band length to obtain $\lambda = 40$ statistical security. For the multi-map, we use 8 byte strings as keys and values following the real-world parameters chosen in prior works [54, 69]. We use SHA256 as the PRF and AES-128-GCM as the encryption scheme.

The comparisons of storage size, query times and setup times can be found in Figure 10. Our experiments confirm that RB-MM results in 16% smaller storage than XorMM. Furthermore, we see that the setup time for RB-MM is significantly faster than XorMM. As an example, RB-MM requires less than 5 seconds for $n = 2^{22}$ whereas XorMM requires almost 8x more time for the same input size. To explain this improvement, we note that XorMM uses algorithms similar to the 3H-GCT OKVS construction [34]. As seen from Figure 6, RB-OKVS is significantly faster during encoding compared to 3H-GCT. The same improvements can also be seen when comparing the setup times of RB-MM and XorMM.

Next, we analyze the query time. Like prior works [54, 69], the query time is independent of the multi-map size n and only depends on the maximum volume ℓ. RB-MM has 5x smaller query times than XorMM that can be seen by the query times

Construction	Online Comm. (MB)			Online Time (s)			Offline Comm. (MB)			Offline Time (s)			Cost ($)		
	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}	2^{16}	2^{18}	2^{20}
KRTW19 [51]	174.99	781.80	3,216.63	29.42	138.06	545.72	0.03	0.03	0.03	0.17	0.17	0.18	0.016	0.071	0.292
GMRSS21 [33]	68.06	292.70	1,525.42	12.70	50.36	286.52	0.03	0.03	0.03	0.28	0.27	0.29	0.006	0.027	0.138
JSZDG22 [43]	101.82	451.97	1,976.60	27.24	144.16	691.23	0.01	0.01	0.01	0.17	0.17	0.23	0.009	0.041	0.180
SKE-PSU [70]	27.73	110.64	-	10.96	36.11	-	11.71	25.58	-	475.11	1,581.52	-	0.005	0.019	-
PKE-PSU [70]	12.27	49.07	184.34	31.65	145.31	552.82	4.69	4.69	4.69	11.38	11.37	11.71	0.002	0.005	0.019
PKE-PSU* [70]	21.92	87.66	350.61	25.95	103.92	402.39	4.72	4.72	4.72	11.69	11.60	12.49	0.003	0.009	0.034
RB-SKE-PSU	27.08	108.02	-	**9.98**	**34.01**	-	11.71	25.58	-	453.46	1,538.18	-	0.005	0.018	-
RB-PKE-PSU	**9.72**	**38.86**	**155.43**	22.73	83.01	333.96	4.69	4.69	4.69	11.23	11.88	11.63	**0.001**	**0.004**	**0.016**
RB-PKE-PSU*	17.04	68.13	272.48	16.22	65.75	**284.28**	4.72	4.72	4.72	11.30	11.75	11.33	0.002	0.007	0.026

Figure 9: Comparison of PSU protocols with 64-bit inputs in 1 Gbits/sec network. Bolded numbers mark the best values.

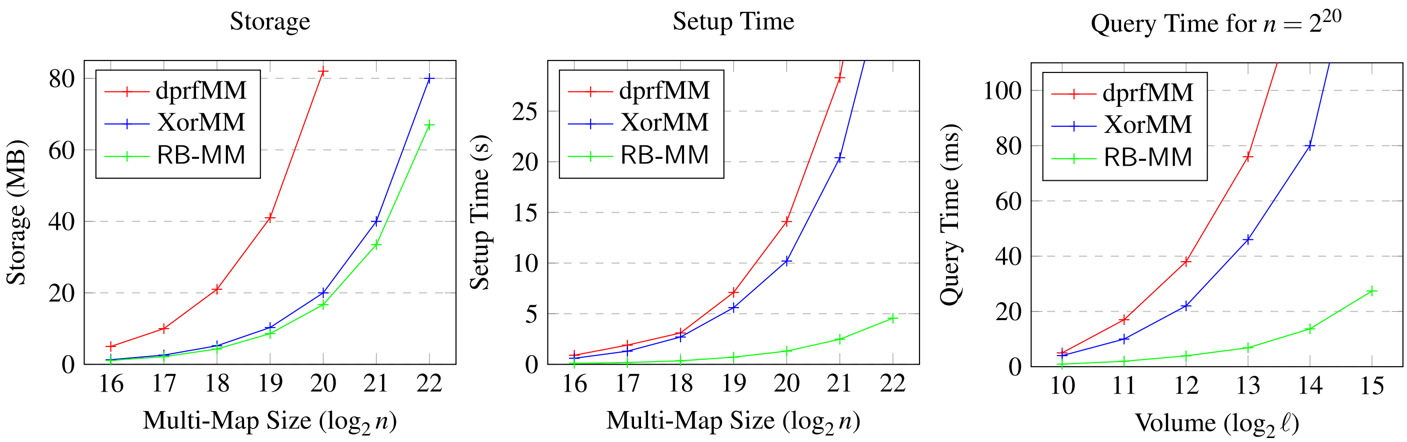

Figure 10: Experiments for storage, query time and setup times for VH-EMMs.

for varying volumes of $\ell \in \{2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}\}$ for a fixed multi-map size of $n = 2^{20}$. Once again, this can be explained due to the efficiency of the decoding of the underlying OKVS. The decoding algorithm of RB-OKVS is much faster than 3H-GCT of which a very close variant is used in XorMM (see Figure 6 for comparisons of decoding times).

The experimental evaluation shows that RB-MM results in the state-of-the-art VH-EMM construction with 16% less storage as well as 5x faster querying and 8x faster setup.

7 Related Work

Oblivious Key-Value Stores. The notion of oblivious key-value stores was introduced by Gamirella *et al.* [34]. However, many prior works such as [25,30,50,56,58,60] implicitly constructed and used OKVS schemes. Recent works [61,70] have explicitly built OKVS schemes with additional properties.

Private Set Intersection. PSI was introduced by Meadows [52] and has been heavily studied in recent years. For some examples, see [22,25,30,34,35,47–50,55–58,60,61,63] as well as references therein. PSI variants studied include circuit PSI [19,39,58,59,64], multi-party PSI [10,18,40,50,53,71], threshold PSI [9,13,36] and unbalanced PSI [23,24,26].

Private Set Union. Kissner and Song [48] presented the first PSU protocol based on polynomials and public-key operation with several follow-ups [27, 32, 37]. PSU using only symmetric-key operations was studied in [12,33,51] leading to a linear scheme [43,70]. Further PSU variants were studied including multi-party [65] and with a third-party helper [16].

Volume-Hiding Encrypted Multi-Maps. The notion of volume-hiding was introduced by Kamara and Moataz [46] and formal definitions were first presented by Patel *et al.* [54]. Several other papers studied VH-EMM including [68, 69]. Volume-hiding with differential privacy was studied in [54] where response size could be independent of maximum volume. Dynamic VH-EMM were studied in [8, 68, 72] where the underlying multi-map may be modified. Finally, verifiable VH-EMM were studied by Wang *et al.* [69].

8 Conclusions

In this work, we present a state-of-the-art OKVS construction, RB-OKVS, that achieves near-optimal rates as high as 0.97 while maintaining efficient encoding and decoding algorithms. Prior works were only able to achieve rates of 0.81 with similar or slower encoding times. Furthermore, RB-OKVS is highly tunable to enable trade-offs between rate and encoding times necessary for various applications. For a variety of rates better than all prior OKVS schemes, RB-OKVS still has the fastest encoding times. Using RB-OKVS, we obtain improved constructions for semi-honest, malicious and circuit PSI, semi-honest PSU and volume-hiding encrypted multi-maps.

References

[1] The difficulty of private contact discovery. https://signal.org/blog/contact-discovery/.

[2] EC2 On-Demand Pricing. https://aws.amazon.com/ec2/pricing/on-demand/.

[3] Helping organizations do more without collecting more data. https://security.googleblog.com/2019/06/helping-organizations-do-more-without-collecting-more-data.html.

[4] mpc4j. https://github.com/alibaba-edu/mpc4j.

[5] Private matching for compute: New solutions to the problem of enabling compute on private set intersections. https://engineering.fb.com/2020/07/10/open-source/private-matching/.

[6] Vole-PSI. https://github.com/Visa-Research/volepsi.

[7] XorMM & VXorMMR. https://github.com/CDSecLab/XorMM.

[8] Ghous Amjad, Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption. *Proceedings on Privacy Enhancing Technologies (to appear)*, 2023.

[9] Saikrishna Badrinarayanan, Peihan Miao, Srinivasan Raghuraman, and Peter Rindal. Multi-party threshold private set intersection with sublinear communication. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 349–379. Springer, Heidelberg, May 2021.

[10] Aner Ben-Efraim, Olga Nissenbaum, Eran Omri, and Anat Paskin-Cherniavsky. PSImple: Practical multiparty maliciously-secure private set intersection. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIACCS 22*, pages 1098–1112. ACM Press, May / June 2022.

[11] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Near-optimal oblivious key-value stores for efficient PSI, PSU and volume-hiding multi-maps. Cryptology ePrint Archive, Paper 2023/903, 2023. https://eprint.iacr.org/2023/903.

[12] Marina Blanton and Everaldo Aguiar. Private and oblivious set and multiset operations. In Heung Youl Youm and Yoojae Won, editors, *ASIACCS 12*, pages 40–41. ACM Press, May 2012.

[13] Pedro Branco, Nico Döttling, and Sihang Pu. Multiparty cardinality testing for threshold private intersection. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 32–60. Springer, Heidelberg, May 2021.

[14] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private matching for compute. Cryptology ePrint Archive, Paper 2020/599, 2020. https://eprint.iacr.org/2020/599.

[15] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas A. Dimitropoulos. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. In *USENIX Security 2010*, pages 223–240. USENIX Association, August 2010.

[16] Ran Canetti, Omer Paneth, Dimitrios Papadopoulos, and Nikos Triandopoulos. Verifiable set operations over outsourced databases. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 113–130. Springer, Heidelberg, March 2014.

[17] Clement Canonne. A short note on poisson tail bounds. http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf, 2017.

[18] Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty PSI and extensions to circuit/quorum PSI. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1182–1204. ACM Press, November 2021.

[19] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-PSI with linear complexity via relaxed batch OPPRF. *Proceedings on Privacy Enhancing Technologies*, 1:353–372, 2022.

[20] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-PSI with linear complexity via relaxed batch OPPRF. *PoPETs*, 2022(1):353–372, January 2022.

[21] Melissa Chase and Seny Kamara. Structured encryption and controlled disclosure. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 577–594. Springer, Heidelberg, December 2010.

[22] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious PRF. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 34–63. Springer, Heidelberg, August 2020.

[23] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

[24] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

[25] Chongwon Cho, Dana Dachman-Soled, and Stanislaw Jarecki. Efficient concurrent covert computation of string equality and set intersection. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 164–179. Springer, Heidelberg, February / March 2016.

[26] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1135–1150. ACM Press, November 2021.

[27] Alex Davidson and Carlos Cid. An efficient toolkit for computing private set operations. In Josef Pieprzyk and Suriadi Suriadi, editors, *ACISP 17, Part II*, volume 10343 of *LNCS*, pages 261–278. Springer, Heidelberg, July 2017.

[28] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. PIR-PSI: Scaling private contact discovery. *Proceedings on Privacy Enhancing Technologies*, 2018(4):159–178, 2018.

[29] Martin Dietzfelbinger and Stefan Walzer. Efficient gauss elimination for near-quadratic matrices with one short random block per row, with applications. In *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[30] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 789–800. ACM Press, November 2013.

[31] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated PSI cardinality with applications to contact tracing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 870–899. Springer, Heidelberg, December 2020.

[32] Keith B. Frikken. Privacy-preserving set union. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 237–252. Springer, Heidelberg, June 2007.

[33] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 591–617. Springer, Heidelberg, May 2021.

[34] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 395–425, Virtual Event, August 2021. Springer, Heidelberg.

[35] Satrajit Ghosh and Tobias Nilges. An algebraic approach to maliciously secure private set intersection. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 154–185. Springer, Heidelberg, May 2019.

[36] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 3–29. Springer, Heidelberg, August 2019.

[37] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. *Journal of Cryptology*, 25(3):383–433, July 2012.

[38] Kyle Hogan, Noah Luther, Nabil Schear, Emily Shen, David Stott, Sophia Yakoubov, and Arkady Yerukhimovich. Secure multiparty computation for cooperative cyber risk assessment. In *2016 IEEE Cybersecurity Development (SecDev)*, pages 75–76. IEEE, 2016.

[39] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

[40] Roi Inbar, Eran Omri, and Benny Pinkas. Efficient scalable multiparty private set-intersection via garbled bloom filters. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 235–252. Springer, Heidelberg, September 2018.

[41] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 370–389. IEEE, 2020.

[42] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS 2012*. The Internet Society, February 2012.

[43] Yanxue Jia, Shi-Feng Sun, Hong-Sheng Zhou, Jiajun Du, and Dawu Gu. Shuffle-based private set union: Faster and more secure. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2947–2964, Boston, MA, 2022. USENIX Association.

[44] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1447–1464. USENIX Association, August 2019.

[45] Seny Kamara and Tarik Moataz. SQL on structurally-encrypted databases. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part I*, volume 11272 of *LNCS*, pages 149–180. Springer, Heidelberg, December 2018.

[46] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 183–213. Springer, Heidelberg, May 2019.

[47] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. Private set intersection for unequal set sizes with mobile applications. *PoPETs*, 2017(4):177–197, October 2017.

[48] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Heidelberg, August 2005.

[49] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[50] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1257–1272. ACM Press, October / November 2017.

[51] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. Scalable private set union from symmetric-key techniques. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 636–666. Springer, Heidelberg, December 2019.

[52] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE, 1986.

[53] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1151–1165. ACM Press, November 2021.

[54] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 79–93. ACM Press, November 2019.

[55] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[56] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[57] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 515–530. USENIX Association, August 2015.

[58] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Heidelberg, May 2019.

[59] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based PSI via cuckoo hashing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 125–157. Springer, Heidelberg, April / May 2018.

[60] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014*, pages 797–812. USENIX Association, August 2014.

[61] Srinivasan Raghuraman and Peter Rindal. Blazing fast psi from improved okvs and subfield vole. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2505–2517, 2022.

[62] Sivaramakrishnan Ramanathan, Jelena Mirkovic, and Minlan Yu. BLAG: Improving the accuracy of blacklists. In *NDSS 2020*. The Internet Society, February 2020.

[63] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1229–1242. ACM Press, October / November 2017.

[64] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Heidelberg, October 2021.

[65] Jae Hong Seo, Jung Hee Cheon, and Jonathan Katz. Constant-round multi-party private set union using reversed laurent series. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 398–412. Springer, Heidelberg, May 2012.

[66] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*, pages 44–55. IEEE, 2000.

[67] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, et al. Protecting accounts from credential stuffing with password breach alerting. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1556–1571, 2019.

[68] Jiafan Wang and Sherman SM Chow. Simple storage-saving structure for volume-hiding encrypted multi-maps. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 63–83. Springer, 2021.

[69] Jianfeng Wang, Shi-Feng Sun, Tianci Li, Saiyu Qi, and Xiaofeng Chen. Practical volume-hiding encrypted multi-maps with optimal overhead and beyond. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2825–2839, 2022.

[70] Cong Zhang, Yu Chen, Weiran Liu, Min Zhang, and Dongdai Lin. Linear private set union from multi-query reverse private membership test. In *USENIX Security 23 (to appear)*, 2023.

[71] En Zhang, Feng-Hao Liu, Qiqi Lai, Ganggang Jin, and Yu Li. Efficient multi-party private set intersection against malicious adversaries. In *Proceedings of the 2019 ACM SIGSAC conference on cloud computing security workshop*, pages 93–104, 2019.

[72] Yongjun Zhao, Huaxiong Wang, and Kwok-Yan Lam. Volume-hiding dynamic searchable symmetric encryption with forward and backward privacy. Cryptology ePrint Archive, Paper 2021/786, 2021. https://eprint.iacr.org/2021/786.

A RB-OKVS Analysis

A.1 Random Band Matrix Proof

It remains to show that RB-OKVS has small error probability and that the solving of random band matrices in the encoding algorithm runs in time $O(nw)$. To do this, we will extend the analysis of random band matrices in [29] for general values of the width parameter w.

As an aside, we note that this proof provides theoretical justification of the low failure probability of RB-OKVS. In practice, we use experimental evaluation to determine the values of $m = (1+\varepsilon)n$ and w in our concrete instantiations. See Section 6.1 for our strategies to instantiate RB-OKVS. Below, we extend the proof in [29] necessary to prove the bound for general values of w.

Lemma 1. *If $m = (1+\varepsilon)n$ for some constant $\varepsilon > 0$ and $w = O(\lambda/\varepsilon + \log n)$, then* RB-OKVS *is an OKVS with error probability at most $2^{-\lambda}$ with encoding time $O(nw)$ for sufficiently large $n = \Omega(\lambda/\varepsilon)$.*

To prove this, we will use the following series of lemmas following the same proof strategy as outlined in [29]. Recall that the encoding algorithm consists of sorting the starting band locations of each row before executing Gaussian elimination on the sorted random band matrix. First, we relate the success probability of the encoding algorithm of RB-OKVS to a variant of hashing that is denoted as coin-flipping Robin Hood hashing (CFRH). In CFRH, we suppose there are n items that are hashed into one of m bins. As a note, we use n items corresponding to the n rows of the random band matrix and the m bins corresponding to the m columns. Each of the n items are assigned to a random bin. The n items are inserted by performing linear probing. That is, checking if a bin is occupied and, if so, moving to the next bin. The main difference in CRFH is that an item is inserted into an empty bin with probability 50% depending on a random coin flip. Otherwise, the item moves onto the next bin in the linear probing process.

First, we define some notation. In the encoding of RB-OKVS, we note that we obtain a sequence of starting band locations $a_1, \ldots, a_n \in [m]$ that yield a sequence of pivots during Gaussian elimination $\mathsf{piv}_1, \ldots, \mathsf{piv}_n \in [m]$. For CFRH, we obtain a sequence of randomly assigned bins $b_1, \ldots, b_n \in [m]$ as well as the final positions of each item $\mathsf{pos}_1, \ldots, \mathsf{pos}_n \in [m]$. Finally, we can define *heights* of each of the m bins in the CFRH as $H_i = |\{j \in [n] \mid b_j \leq i < \mathsf{pos}_j\}$. In words, H_i represents the number of items that actually probe the number of the i-th bin without being placed into the bin. This will be useful to bound the total running time of the encoding time of RB-OKVS.

Immediately, we can see that the starting band locations and pivots correspond to the random bin assignments and final positions. In particular, we can obtain a modified version of the following lemma from [29]:

Lemma 2. *The following three properties are true:*

- RB-OKVS *encoding succeeds if and only if CFRH succeeds. On success, we get that $\mathsf{piv}_i = \mathsf{pos}_i$ for all $i \in [n]$.*
- *A successful run of* RB-OKVS *encoding performs at most $\sum_{j \in [m]} H_j$ row additions.*
- *Conditioned on $\max_{j \in [m]} H_j \leq w - 2\lambda - \log n$ being true, the encoding algorithm of* RB-OKVS *succeeds except with probability $2^{-2\lambda}$.*

Proof. The first two properties follow immediately from Lemma 3 in [29]. Therefore, we only prove the final property. Consider the i-th item that is inserted. Note that all items are inserted in increasing choices of bins b_i. As $H_i \leq w - 2\lambda - \log n$, we know that at least $2\lambda + \log n$ of the next w bins are unoccupied. Therefore, the probability that the i-th item is not inserted into any of these bins is $2^{-2\lambda - \log n}$. By a Union bound over all n items, we see the probability that one item is not inserted into any of the m bins is $2^{-2\lambda}$. □

Next, we consider a Poisson approximation. At a high level, the goal is to replace the real CFRH process with an approximation where the number of items that are assigned to the i-th bin is drawn from a Poisson distribution. Formally, we draw z_i from the Poisson distribution with parameter $(1+\varepsilon)$. Note, this is expected to be larger than the standard process where each bin will be assigned $n/m \leq 1$ items in expectation. Let H_i' be the resulting heights of the Poisson approximated version of CFRH. Then, we prove the following:

Lemma 3. *There is a coupling between an ordinary run of CFRH (with H_i) and a Poissonised run with (H_i') such that we have $H_i' \geq H_i$ for all $i \in [m]$ except with probability $2^{-2\lambda}$ for sufficiently large $n = \Omega(\lambda/\varepsilon)$.*

Proof. The coupling proof follows identically from Lemma 4 in [29]. The only difference is that we need to prove that the coupling succeeds with probability exponentially small in λ. To do this, we must bound the probability of the event that the sum of the Poisson variables is less than m. For this, we can use known concentration bounds for Poisson distributions. In particular, we use the following bound in [17]:

$$\Pr[X \leq \mu - x] \leq e^{-\frac{x^2}{2(\mu+x)}}$$

where X is a Poisson variable with parameter μ. It suffices to plug in $x = \varepsilon \cdot n$ to get that the event occurs with probability at most $e^{-(\varepsilon^2 \cdot n)/(2+6\varepsilon)}$. Therefore, if $2\lambda \leq \varepsilon^2 \cdot n/(2+6\varepsilon)$, we get this occurs with probability at most $2^{-2\lambda}$. □

The benefit of the above coupling is that we can model the heights of each bin as a Markov chain. In particular, consider the i-th bin. Then, the height is $H_i' = H_{i-1}' + z_i - 1$ assuming an item is placed into the i-th bin. We can use a variable g_i that is a random variable from a Geometric distribution with probability $1/2$. We define $b_i = \mathbf{1}_{g_i > H_{i-1}' + k_i}$ to determine whether no item is placed into the i-th bin and get that $H_i' = H_{i-1}' + z_i - 1 + b_i$.

Afterwards, we utilize another coupling where we replace the geometrically distributed random indicator b_i by using a Poission random variable with slightly larger mean. In particular, we use the following:

$$X_0 = 0 \text{ and } X_i = \max(0, X_{i-1} + z_i' - 1)$$

where z_i' is a Poisson variable with parameter $(1+2\varepsilon)$. We obtain the coupling following directly from Lemma 5 in [29].

Lemma 4. *There is a coupling between $\{X_i\}_{i \in [m]}$ and $\{H_i\}_{i \in [m]}$ such that $X_i + \log(4/\varepsilon) \geq H_i'$.*

Finally, we use facts from queuing theory to complete the proof. In particular, one can view the Markov chain above as a M/D/1 queue. Using known facts about M/D/1 queues, we can complete the proof of the main lemma.

Proof of Lemma 1. First, we use Lemma 4 to analyze the success of RB-OKVS encoding using the heights X_i that stochastically dominate the original heights H_i except with probability $2^{-2\lambda}$. By Fact 1(ii) in [29], we know that $\Pr[X_i > w/2] \leq 2^{-2\lambda} \cdot m^{-1}$ assuming $w = \Theta(\lambda/\varepsilon + \log n)$ and using the fact that $m = \Theta(n)$. By a Union bound over all m heights, we get that all heights $\{X_i\}_{i \in [m]}$ are at most $w/2$ except with probability $2^{-2\lambda}$. Finally, we can plug this into Lemma 2 to get that the encoding algorithm of RB-OKVS succeeds with probability at most $3 \cdot 2^{-2\lambda} \leq 2^{-\lambda}$. Therefore, the error probability of RB-OKVS is at most $2^{-\lambda}$.

Next, we analyze the running time of the encoding algorithm of RB-OKVS. We use Lemma 2 where we know that the number of row additions of the encoding algorithm of RB-OKVS is exactly $\mathsf{E}[H_1 + H_2 + \ldots + H_m]$. By linearity of expectation, it suffices to analyze the expectation of a single height as $\mathsf{E}[H_i] \leq \mathsf{E}[X_i + \log(4/\varepsilon)] = 1 + 2\varepsilon + \log(4/\varepsilon) = O(1)$. Therefore, the total expected running time is $O(nw)$ as each row addition requires $O(w)$ time. □

A.2 RB-OKVS Proof

Proof of Theorem 1. From Lemma 1, we immediately get the claims that RB-OKVS has error probability at most $2^{-\lambda}$ and encoding time at most $O(nw)$. We note that decoding requires performing a dot product where one vector has at most w non-zero entries using only $O(w)$ time.

Secondly, we prove that RB-OKVS satisfies the oblivious and doubly oblivious properties. To do this, we first prove that RB-OKVS is doubly oblivious. Recall that all free variables during Gaussian elimination are chosen uniformly at random. Back substitution solves for the *lead* variables s_i of each row j (i.e., the first non-zero entry in each row j) in terms of previously chosen $s_{i'}$, $i' > i$, and some subset $\{v_{j_1}, \ldots, v_{j_k}\} \subseteq \{v_1, \ldots, v_n\}$, containing v_j. Since, v_j is chosen uniformly at random, s_i will be distributed uniformly, regardless of all other values $s_{i'}$, $i' > i$, and $v_{j_l} \neq v_j$. Thus, it is clear that this construction satisfies double obliviousness. Finally, we note that RB-OKVS being doubly oblivious immediately implies that RB-OKVS is oblivious. As the output encoding is a uniformly random element, no adversary can distinguish between two different output encodings.

Next, we show that RB-OKVS satisfies the random decodings properties. First, we leverage the fact that RB-OKVS is already doubly oblivious. Therefore, each output of the encoding is already a uniformly random field element. For decoding any single key, we note that we are taking the dot product of a random w-bit binary string and a subset of w uniformly random field elements from the encoding. As long as the binary string is not all zero, then the output is a uniformly random element. Therefore, we get that decoding outputs are random decodings except with probability 2^{-w}, which is the probability that the random w-bit string is all zero.

Finally, we prove that RB-OKVS satisfies the additional OKVS properties needed by various applications of being binary and linear. We note that the decoding algorithm consists of taking the sum of at most w elements of the encoding satisfying the binary property. In particular, this also immediately implies that RB-OKVS is also linear as the binary property is a special case of the linearity property. □

Parameters	**Best fit line $\lambda = aw + b$**
$\varepsilon = 0.03$	
$n = 2^{10}$	$0.08047w - 3.464$
$n = 2^{14}$	$0.08253w - 5.751$
$n = 2^{16}$	$0.08241w - 7.023$
$n = 2^{18}$	$0.08192w - 8.569$
$n = 2^{20}$	$0.08313w - 10.880$
$n = 2^{24}$	$0.08253w - 14.671$
$\varepsilon = 0.05$	
$n = 2^{10}$	$0.1388w - 4.424$
$n = 2^{14}$	$0.1389w - 6.976$
$n = 2^{16}$	$0.1399w - 8.942$
$n = 2^{18}$	$0.1388w - 10.710$
$n = 2^{20}$	$0.1407w - 12.920$
$n = 2^{24}$	$0.1376w - 16.741$
$\varepsilon = 0.07$	
$n = 2^{10}$	$0.1947w - 5.383$
$n = 2^{14}$	$0.1926w - 8.150$
$n = 2^{16}$	$0.1961w - 10.430$
$n = 2^{18}$	$0.1955w - 12.300$
$n = 2^{20}$	$0.1939w - 14.100$
$\varepsilon = 0.1$	
$n = 2^{10}$	$0.2747w - 6.296$
$n = 2^{14}$	$0.2685w - 9.339$
$n = 2^{16}$	$0.2740w - 11.610$
$n = 2^{18}$	$0.2715w - 13.390$
$n = 2^{20}$	$0.2691w - 15.210$
$n = 2^{24}$	$0.2751w - 19.830$

Figure 11: Best fit lines generated in the experimental evaluation from different values of ε and n.

B Best Fit Lines for RB-OKVS

We present the best fit lines that we obtained using linear regression from our experimental evaluation in Section 6.1 for reference and future usage. See Figure 11 for best fit lines.