Blazing Fast PSI from Improved OKVS and Subfield VOLE

Peter Rindal, Srinivasan Raghuraman

No Institute Given

Abstract. We present new semi-honest and malicious secure PSI protocols that outperform all prior works by several times in both communication and running time. For example, our semi-honest protocol for $n = 2^{20}$ can be performed in 0.37 seconds compared to the previous best of 2 seconds (*Kolesnikov et al., CCS 2016*). This can be further reduced to 0.16 seconds with 4 threads, a speedup of 12×. Similarly, our protocol sends $187n$ bits compared to $426n$ bits of the next most communication efficient protocol (*Rindal et al., Eurocrypt 2021*). These performance results are obtained by two types of improvements.

The first is an optimization to the protocol of Rindal et al. to utilize *sub-field* vector oblivious linear evaluation. This optimization allows our construction to be the first to achieve a communication complexity of $\mathcal{O}(n\lambda + n \log n)$ where λ is the *statistical* security parameter. In particular, the communication overhead of our protocol does not scale with the computational security parameter times n.

Our second improvement is to the OKVS data structure which our protocol crucially relies on. In particular, our construction improves both the computation and communication efficiency as compared to prior work (*Garimella et al., Crypto 2021*). These improvements stem from algorithmic changes to the data structure along with new techniques for obtaining both asymptotic and tight concrete bounds on its failure probability. This in turn allows for a highly optimized parameter selection and thereby better performance.

1 Introduction

1.1 Private-Set Intersection

In this work, we present new improvements for efficiently performing private set intersection (PSI). Here, two mutually distrusting parties, each hold a set of values X, Y respectively and wish to learn the intersection between their sets $X \cap Y$ without revealing any additional information. In particular, the first party (receiver) with set X should not learn anything about $X \setminus Y$ beyond the size of Y. Similarly, the second party (sender) should not learn any information about X beyond the size of it.

PSI protocols date back to the 1980s [Mea86] and were initially based on OPRFs/Diffie–Hellman. In fact, several modern protocols [CT10; IKN+20; BKM+20]

are still based on the protocol of [Mea86]. With the invention of oblivious transfer extension [IKNP03], research into a new protocol family was initiated by Schneider et al. [PSSZ15] along with many derivatives [PSZ14; KKRT16; RR17; OOS17; PRTY19]. These protocols offer improved efficiency at the expense of increased communication.

However, with the advent of [PRTY20] and the closely related protocols of [RS21; GPR+21], the situation has begun to change. [PRTY20] introduced a new data structure called an OKVS which offers a very convenient way to represent the sets X and Y. [PRTY20] went on to combine their OKVS and the PSI protocol of [PRTY19] to obtain one of the most efficient PSI protocols of their time. Shortly after, [RS21] was able to observe that the OKVS data structure can be more efficiently combined with vector oblivious linear evaluation (VOLE)[BCG+19; CRR21] to obtain an even more efficient PSI protocol. Concurrently, [GPR+21] proposed an improvement to OKVS which allowed for an improved communication overhead of the [PRTY20] PSI protocol. Despite all these advances, the non-OKVS based protocol of [KKRT16] remains the most computationally efficient while [RS21] is the most communication efficient, and [GPR+21] is a compromise between the two.

We make the observation that recent improvements to VOLE [CRR21] and OKVS [GPR+21] can be applied to [RS21]. While this does further reduce the communication complexity of [RS21], the running time of the protocol remains slower than [KKRT16] due to the computational overhead of the [GPR+21] OKVS data structure. Looking forward, we will present significant improvements to the OKVS data structure of [PRTY20; GPR+21] along with new techniques for further reducing the communication overhead of [RS21].

We further note that our improvements also translate to several other PSI related functionalities. To name a few, circuit PSI[HEK12; PRTY19; RS21] is a functionality similar to normal PSI except that the intersection is output to the parties in secret shared form. This can be particularly useful when additional encrypted computation needs to be performed on the intersection before revealing it, e.g. compute the cardinality or the sum of associated values[IKN+20]. Alternatively, the recent multi-party PSI protocol of [NTY21] makes use of both OKVS and a primitive known as an OPRF/OPPRF which can, in turn, be based on [RS21] along with our improvements. We leave the application of our work to these and other protocols as future work.

1.2 Oblivious Key-Value Stores

The aforementioned data structure known as a Oblivious Key-Value Stores (OKVS) consists of algorithms Encode and Decode. Encode takes a list of key-value pairs $(k, v) \in L$ as input and returns an abstract data structure S which encodes L. Decode takes such a data structure and a key k' as input, and gives some output v'. Decode can be called on any key, but if it is called on some k' that was used to generate S, then the result is the corresponding v'. An OKVS scheme is said to be *linear* if for any k, $\mathsf{Decode}(S, k)$ can be expressed as a linear

combination of S, i.e. $\mathsf{Decode}(S, k_i) = \langle S, \mathsf{row}(k_i)\rangle = v_i$ where row is some public function. In this work we restrict ourselves to linear OKVS schemes.

In PaXoS [PRTY20], n items are encoded into a vector S of length $m = 2.4n$. This scheme is parameterized by $\mathsf{row} : \{0,1\}^* \to R$ with $R \subset \{0,1\}^m$ containing only weight 2 vectors. Decoding of a key k is therefore the sum $S_i + S_j = \langle S, \mathsf{row}(k)\rangle$ where $\mathsf{row}(k)$ is 1 at positions i, j. Encoding is done by generating the *cuckoo graph* implied by the n keys and row. In that graph, there are m vertices $u_1, \ldots, u_m$ such that each $\mathsf{row}(k)$ implies an edge (u_i, u_j). The encoder then *peels* that graph, by recursively removing each edge (u_i, u_j) for which the degree of either u_i or u_j is 1, and pushing that (k, v) pair to a stack. Let us assume that all edges have been removed in this way. Then, the unpeeling process iteratively pops the next (k, v) pair from the stack and uses it to fill the vector's entries: S_i and/or S_j will be unassigned and then assigned a value such that $S_i + S_j = v$.

However, there is a high probability that the peeling process cannot remove all edges from the graph due to all remaining vertices having degree greater than 1. The size of the remaining graph is known to be at most $\mathcal{O}(\log n)$ vertices. PaXoS solves this issue by changing the distribution of row such that it outputs a uniformly random weight 2 vector of length $2.4n$ followed by $\mathcal{O}(\log n) + \lambda$ randomly sampled bits, i.e. $m = 2.4n + \mathcal{O}(\log n) + \lambda$. Intuitively, this allows the encoder solve the system of equations corresponding to these last $\mathcal{O}(\log n)$ vertices using the last $\mathcal{O}(\log n) + \lambda$ random bits while all the other vertices are solved using the peeling algorithm above.

[GPR+21] proposed generalizing the above construction to have the main part of $\mathsf{row}(k)$ output weight 3 vectors as opposed to weight 2. Indeed, it is well-known that the efficiency of cuckoo hashing improves significantly when using three rather than two hash functions [Wal21]. [GPR+21] proposes that m can be reduced to $m = 1.3n + \mathcal{O}(\log n) + \lambda$ which is a reduction of almost $2\times$. Due to a lack of techniques for bounding of the failure probabilities of their primary construction, [GPR+21] presents methods for amplifying the probabilistic guarantees of an OKVS. They show how to use an OKVS with failure probability p to build an OKVS with failure probability $c \cdot p^d$ (for explicit constants c, d). In particular, [GPR+21] empirically verifies that their main construction achieves a failure probability of at most 2^{-29} and then amplifies this to be arbitrarily small, i.e. $2^{-\lambda}$, at a moderate cost to running time and compactness.

1.3 Our Contributions

In this work, we propose an improved OKVS construction and use it as a building block in order to obtain the fastest and most communication efficient PSI protocols to date.

- We put forth a framework to both theoretically (asymptotically) and empirically (concretely) understand the tight concrete bounds of the failure probabilities associated with the randomized OKVS construction based on cuckoo hashing involving two or more hash functions, without the need to resort to the amplification techniques put forth in [GPR+21].

- Equipped with the understanding from above, we are able to come up with a more efficient OKVS construction which improves considerably over the state of the art in terms of size and speed. In particular, in the range of parameters we work in, our OKVS construction is "optimal" with overwhelming probability.
- Finally, we obtain the most efficient PSI protocol to date, by employing our improved OKVS in addition to a new optimization we introduce to the OPRF/PSI construction put forth in [RS21]. These improvements enable our construction to perform a PSI of $n = 2^{20}$ items in 0.37 and 0.16 seconds in the single and multi threaded settings, respective. On the same hardware, the previous fastest protocol required 2 seconds, a $10\times$ improvement. Moreover, our new optimizations to the protocol of [RS21] achieves the best communication complexity to date, performing a PSI of n items with $\mathcal{O}(n\lambda + n \log n)$ bits of communication, where λ is the *statistical* security parameter. For example, a PSI with $n = 2^{20}$ can be performed with $157n$ bits of communications compared to the previous best of $426n$ [RS21].

1.4 Notation

We use κ as the computational security parameter and λ for statistical security. $[a, b]$ denotes the set $a, a + 1, \ldots, b$ and $[b]$ is shorthand for $[1, b]$. We denote row vectors $\vec{A} = (a_1, \ldots, a_n)$ using the arrow notation while the elements are indexed without it. A set $S = \{s_1, \ldots, s_n\}$ will use similar notation. For a matrix M, we use $\vec{M}_i$ to denote its ith row vector, and $M_{i,j}$ for the element at row i and column j. $\langle \vec{A}, \vec{B} \rangle$ denotes the inner product of $\vec{A}$,$\vec{B}$. We use $=$ to denote the statement that the values are equal. Assignment is denoted as $:=$ and for some set S, the notation $s \leftarrow S$ means that s is assigned a uniformly random element from S. If a function F is deterministic then we write $y := F(x)$ while if F is randomized we use $y \leftarrow F(x)$ to denote $y := F(x; r)$ for $r \leftarrow \{0, 1\}^*$.

2 Our OKVS Construction

We begin by describing our core OKVS construction which maps a set of key-value pairs $\{(k_1, v_1), ..., (k_n, v_n)\}$ to a vector $P \in \mathbb{F}^m$. The formal description of the construction can be found in Figure 1. Let us define[1] $\mathsf{row}(k, r) := \mathsf{row}'(k, r)||\hat{\mathsf{row}}(k, r) \in \mathbb{F}^m$ where $\mathsf{row}'(k, r) \in \{0, 1\}^{m'}$ outputs a uniformly random (sparse) weight w vector and $\hat{\mathsf{row}}(k, r) \in \mathbb{F}^{\hat{m}}$ outputs a short dense vector. The exact distribution of $\hat{\mathsf{row}}$ will vary and is discussed below. At a high level, our algorithm begins by sampling an instance key $r \leftarrow \{0, 1\}^\kappa$ and defining

$$H := \begin{bmatrix} \mathsf{row}(k_1, r) \\ \ldots \\ \mathsf{row}(k_n, r) \end{bmatrix} \in \mathbb{F}^{n \times m}.$$

[1] For technical reasons row must be randomized which we explicitly perform using r.

The output will be a vector $P \in \mathbb{F}^m$ such that

$$HP = (v_1, ..., v_n)$$

The algorithm will first perform *triangulation* which permutes the rows and columns of H. In particular, it defines two permutation matrices $\pi^{\mathsf{r}} \in \{0,1\}^{n \times n}, \pi^{\mathsf{c}} \in \{0,1\}^{m \times m}$ over the rows and columns of H, respectively. Let T correspond to applying the permutations to H.

$$T := \pi^{\mathsf{r}} \cdot H \cdot \pi^{\mathsf{c}} = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}.$$

These permutations will have the structure that for some *gap* $g \leq n$, T can be decomposed into $A, ..., F$ such that $F \in \{0,1\}^{\delta \times \delta}$ where $\delta := n - g$ is lower triangular with ones on the diagonal. Additonally, $B \in \mathbb{F}^{g \times \hat{m}}, E$ will consist of the dense columns generated by $\hat{\mathsf{row}}$. Our *triangulation* algorithm will aim to minimize the g subject to F being lower triangular. The next phase computes

$$T' := \begin{bmatrix} I & -CF^{-1} \\ 0 & I \end{bmatrix} \cdot T = \begin{bmatrix} A' & B' & 0 \\ D & E & F \end{bmatrix}$$

where $A' := -CF^{-1}D + A, B' := -CF^{-1}E + B$. Let $B^* := QB'$ be the (lower) reduced row echelon form of B'. If B^* does not have full row rank, e.g. $g > \hat{m}$, the algorithm aborts. Otherwise, let

$$T^* := \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} \cdot T' = \begin{bmatrix} A^* & B^* & 0 \\ D & E & F \end{bmatrix}$$

where both B^*, F are lower triangular with ones on their diagonal. We can apply the same row transformation to $V := (v_1, ..., v_n)$ to obtain

$$V^* := \begin{bmatrix} Q & -QCF^{-1} \\ 0 & I \end{bmatrix} \cdot \pi^{\mathsf{r}} \cdot V$$

Since T^* itself is lower triangular, we can compute

$$\begin{aligned} T^* \cdot P^* &= V^* \\ P^* &= T^{*-1} \cdot V^* \end{aligned}$$

in linear time using *back-substitution* by solving one row at a time in a bottom up manner. Finally, the algorithm output $P := P^* \cdot (\pi^{\mathsf{c}-1})^{\intercal}$.

The running time of this algorithm is $\mathcal{O}(g^2 + m)$ plus the time required to compute the $\pi^{\mathsf{r}}, \pi^{\mathsf{c}}$ permutations. Therefore, the efficiency of this algorithm depends on the triangularization phase minimizing g. Moreover, the algorithm aborts if B^* (equivalently B') does not have full row rank (i.e. the rows of B^* are linearly dependent).

Triangulation Our triangulation algorithm runs in time $\mathcal{O}(m)$. We prove that this algorithm is optimal for the case of $g = 0$ and near optimal with overwhelming probability for the small values of g which we will be concerned about.

The algorithm takes as input the matrix H and outputs two permutations $\pi^{\mathsf{r}}, \pi^{\mathsf{c}}$ which permute the rows and columns of H respectively. These permutations will place H in g-approximate lower triangular form. The algorithm proceeds by iteratively selecting the next column with "low weight" and moving it to the right to be part of F. When a column is selected, one of its non-zero rows is *assigned* to the diagonal (move down). Any additional non-zero rows, which have not previously been assigned, will be made part of the gap. In this way, at each iteration we add one row/column to F which extends the diagonal while zero or more rows are added to C.

In Section 4.3 we prove that this algorithm is optimal for $g = 0$ and performs well for small g. Moreover, in Section 4.2 we note that the general case for $g = \mathcal{O}(n)$ is likely intractable to solve optimally. For our construction this turns out to not be an issue due to g being small with overwhelming probability, e.g. $g < 4$, as shown in Section 4.

We note that in the special case that we do have an upper bound $\hat{g}$ on g, it is possible to optimally triangulate H. At a high level, one would simply try all possible ways of placing H in $\leq \hat{g}$-approximate lower triangular form as described above by trying all possible sets of at most $\hat{g}$ rows of H being made part of the gap. However, naïvely, the complexity of this operation would be $\mathcal{O}(n^{\hat{g}})$. We can do much better than this. As we shall see later, one need only try all possible sets of at most $\hat{g}$ rows of the so-called 2-*core of* G_H (G_H is a hypergraph defined by H), something we will define later in Section 4. This modified triangulation algorithm is described in Figure 12. Let h_2 denote the size of the 2-core of G_H. Then, the complexity of the modified triangulation step would be $\mathcal{O}(m + h_2^{\hat{g}})$. As we will describe in Section 4, in the parameter regime we consider, $\hat{g} = \mathcal{O}(1)$ and $h_2 \leq c \cdot \hat{g}$ with overwhelming probability for $c = \mathcal{O}(1)$. In fact, empirically, $c \approx 2$ with overwhelming probability and $\hat{g}$ is a small constant (for instance, between 2 and 5). The proofs of all of the above facts and the optimality of the modified triangulation can be found in Section 4.

Full row rank of B' A key requirement of our algorithm is that B' contains an invertible submatrix of size g. A trivial requirement for this is that m', the number of columns contained in B', is greater than or equal to g. This is ensured by upper bounding g as $\hat{g}$ and setting m' accordingly. However, it is still possible that B' does not have full row rank even if $m' \geq \hat{g}$. The key factors that determine this are the distribution of $\mathsf{r\hat{o}w}$ and how large $m' - g$ is. We present two techniques for ensuring that B' has full row rank with overwhelming probability.

Binary $\mathsf{r\hat{o}w}$*.* The first technique was directly inspired by [PRTY20]. Given the upper bound $\hat{g}$ on the gap size g, $\mathsf{r\hat{o}w}$ is defined as a uniform function $\mathsf{r\hat{o}w} : \{0,1\}^* \to \mathbb{F}^{m'}$ where $\mathbb{F} := \{0,1\}$, $m' = \hat{g} + \lambda$ and λ is the statistical security parameter. Given the distribution of $\mathsf{r\hat{o}w}$, we can bound the probability

Parameters: Statistical and computational security parameters λ, κ respectively. Input length n with elements $(z_i, v_i) \in \mathcal{Z} \times \mathcal{V}$, a finite field $\mathbb{F}$, a finite group $\mathbb{G}$. For some $m' = \mathcal{O}(n), \hat{m} = \mathcal{O}(\lambda)$, let the output length be $m = m' + \hat{m}$.

Encode $((z_1, v_1), ..., (z_n, v_n); r)$:

1. **[Sample]** Sample[a] random functions $\mathsf{row}' : \mathcal{Z} \to \mathcal{S}_w$ and $\hat{\mathsf{row}} : \mathcal{Z} \to \mathbb{F}^{\hat{m}}$ where $\mathcal{S}_w \subset \{0,1\}^{m'}$ is the set of all weight w strings. Let $\mathsf{row}(z) := \mathsf{row}'(z) || \hat{\mathsf{row}}(z)$ for all $z \in \mathcal{Z}$ and define
$$H := \begin{bmatrix} \mathsf{row}(z_1) \\ ... \\ \mathsf{row}(z_n) \end{bmatrix} \in \mathbb{F}^{n \times m}$$
2. **[Triangulate]** Let $H' := H, J := \emptyset$. While H' has rows:
   (a) Select $j \in [m]$ such that the jth (sparse) column of H' has the minimum non-zero weight.
   (b) Append index j to the ordered list J. Remove all rows $i \in [n]$ from H' for which $H'_{i,j} \neq 0$.
   Define $\delta := |J|$, the gap as $g := n - \delta$, permutation matrices $\pi^{\mathsf{r}} \in \{0,1\}^{n \times n}, \pi^{\mathsf{c}} \in \{0,1\}^{m \times m}$ such that ${\pi^{\mathsf{c}}}_{m-k, m-\delta-k} = 1$ for $k \in [0, \hat{m})$, ${\pi^{\mathsf{c}}}_{J_i, m+1-i} = 1$ and ${\pi^{\mathsf{r}}}_{n+1-i, i'} = 1$ for some i' where $H_{i', J_i} \neq 0$ and all $i \in [\delta]$. Let
$$T := \pi^{\mathsf{r}} \cdot H \cdot \pi^{\mathsf{c}} = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}.$$
   where $F \in \{0,1\}^{\delta \times \delta}$ is lower triangular, $B \in \mathbb{F}^{g \times \hat{m}}, E \in \mathbb{F}^{\delta \times \hat{m}}$ are the dense columns.
3. **[Zero-C]** Compute
$$T' := \begin{bmatrix} I & -CF^{-1} \\ 0 & I \end{bmatrix} \cdot T = \begin{bmatrix} A' & B' & 0 \\ D & E & F \end{bmatrix}$$
4. **[Solve-Dense]** If B' does not have full row rank, abort. Otherwise, let $B^* := QB'$ be the (lower) reduced row echelon form of B', and
$$T^* := \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix} \cdot T' = \begin{bmatrix} A^* & B^* & 0 \\ D & E & F \end{bmatrix}, \qquad v^* := \begin{bmatrix} Q & -QCF^{-1} \\ 0 & I \end{bmatrix} \cdot \pi^{\mathsf{r}} \cdot v$$
5. **[Back-propagation]** Compute
$$\vec{P}^* := {T^*}^{-1} \cdot v^*$$
   via back propagation and return $\vec{P} := \vec{P}^* {\pi^{\mathsf{c}}}^{-1}$.

Decode $(\vec{P}, z, r)$:

1. Sample $\mathsf{row} : \mathcal{Z} \to \mathbb{F}^m$ using randomness r as was done in Encode.
2. Return $\langle \mathsf{row}(z), \vec{P} \rangle$.

[a] Sample $\mathsf{row}', \hat{\mathsf{row}}$ using randomness r.

Fig. 1: OKVS scheme.

of B' having full row rank as $1 - 2^{-\lambda}$. First observe the $B' = -CF^{-1}E + B$ is uniformly random since B is. A random $m' \times \hat{g}$ binary matrix has full row rank with probability

$$\frac{\Pi_{i=0}^{\hat{g}-1}(2^{m'} - 2^i)}{2^{m'\hat{g}}} \geq 1 - 2^{\hat{g}-m'} = 1 - 2^{-\lambda}$$

Here, the ith term of the numerator is the probability that the ith row is linearly independent of the previous rows while the denominator is the total number of such matrices.

Field $\hat{\mathsf{row}}$. While the previous approach works, it has the downside of significantly increasing the weight of the row function. This imposes a non-trivial performance overhead due the need for the additional (binary) multiplications. As an alternative, we propose defining $\mathbb{F}$ as a large field, e.g. with order 2^κ, and defining $\hat{\mathsf{row}}(k, r) = (\hat{k}, \hat{k}^2, ..., \hat{k}^{m'})$ where $m' := \hat{g}$ is an upper bound on the size of the gap, $\hat{k} := I(k, r)$, and $I : \{0,1\}^* \to \mathbb{F}$ is a random mapping. In this case, $B' = -CF^{-1}E + B$ will have full row rank with overwhelming probability. This is because B is a random Vandermonde matrix (which has full row rank iff the $\hat{k}$ values are distinct) and the argument to show that B has full row rank with overwhelming probability readily extends to show that the sum of a fixed matrix (here, $-CF^{-1}E$) and B will also have full row rank with overwhelming probability. Recall that we can argue that a random Vandermonde matrix will have full row rank as follows. Consider the sub-matrix formed by the first n columns of the matrix (where n is the number of rows). We will show that this matrix has full column rank, which completes the proof. Suppose there is some linear combination of the columns that yields the zero vector. This then translates to the fact that the Vandermonde polynomial of degree $n - 1$ (or n) has n distinct (with overwhelming probability) roots (other than 0), which is possible iff the polynomial is the zero polynomial. This argument can be extended to show that the sum of a fixed matrix and a random Vandermonde matrix will also have full row rank with overwhelming probability. As before, a column combination can be expanded to a polynomial of degree n which admits at most n distinct roots in $\mathbb{F}$. Since the Vandermonde matrix is randomly sampled, with overwhelming probability, the value sampled will be a non-root of the polynomial under consideration[2]. The main downside of this approach is the need for performing multiplication over a large field. However, due to the hardware support for $\mathbb{F}_{2^{128}}$ multiplication, the performance of this approach significantly outperforms the binary case. One limitation of this approach is that the value domain $\mathcal{V}$ must be a vector over the large field while the binary approach can encode any string.

3 Clustering

Although the aforementioned construction achieves linear time encoding for the desired parameter regime, it has a major limitation when encoding/decoding

[2] Note that we assume r is sampled after the k values are fixed.

very large input lengths, e.g. $n = 2^{20}$. The bulk of the memory accesses are effectively random access into an array of size $\mathcal{O}(n)$. When n is sufficiently large, the CPU cache can no longer hold the required data and it must be fetched from main memory. This can impose a several times slowdown in the running time.

To mitigate the effect of this, we propose restricting the distribution of $\mathsf{row}' : \{0,1\}^* \to S_w$ such that the non-zero positions are clustered together. In particular, we redefine $S_w \subset \{0,1\}^{m'}$ to be the set of all w weight strings such that for all $s \in S_w$, $\mathit{support}(s) \subset (im^*, im^* + m^*]$ for some $i \in [0, \beta), m^*$ where β is the number of clusters and m^* is the cluster size. That is, each string $s \in S_w$ has all of its non-zero positions clustered in the ith length m^* non-overlapping substring, for some i. Each cluster can then be triangulated independently, where only one cluster needs to be processed at a time. In our implementation we set $m^* \approx 2^{14}$ which gives a good tradeoff between parameters.

While clustering has advantages, it can also impact the failure probability of the encoding algorithm. To address this, we modify the distribution of $\hat{\mathsf{row}}$. In doing this the core issue is that each cluster will result in some gap with a probability based on m^* and how many inputs were mapped to this cluster. Given an upper bound n^* on the number of inputs items mapped to any cluster, one can then determine an upper bound on the maximum gap size $\hat{g}$ for any cluster. We then consider two strategies for ensuring that the gap can be solved using our algorithm.

Combined. $\hat{\mathsf{row}}$ can be defined as a uniform function $\hat{\mathsf{row}} : \mathcal{Z} \to \mathbb{F}_2^{\hat{m}^*}$ where $\hat{m}^* := \lambda + \hat{g}\beta$. In this formulation we leverage the fact that the λ extra columns can be shared among the β clusters. Since λ is likely several factors larger than $\hat{g}$, this will significantly reduce the weight of the rows. Overall this approach allows for a relatively compact encoding while still enabling clustering.

Separate. Alternatively, $\hat{\mathsf{row}}$ can be defined to similarly have β clusters which are correlated with row'. In particular, if $\mathsf{row}'(z)$ is mapped to the ith cluster, then $\hat{\mathsf{row}}(z)$ should output a uniform random string in $\mathbb{F}^{\hat{m}\beta}$ subject to all but the ith non-overlapping substring of length $\hat{m}$ being zero. In this way we are effectively creating β independent instances of the underlying OKVS. This approach is slightly less compact compared to the previous but allows for greater independence between the clusters and a lower row weight.

Overall, clustering allows the algorithm to process chunks of size m^* which can be orders of magnitude smaller than m. This improves the memory locality and cache efficiency. Moreover, say for example we ensure that $m^* = 2^{16}$. It is then the case that all internal state values can be stored in a 16-bit integer, as compared to say 32 or 64-bit, which allows the implementation to have even better memory locality, e.g. by a factor of 2 or 4.

Additionally, clustering naturally lends itself to a multi-threaded implementation. Items can be mapped to clusters and then a thread can process each cluster without the need for expensive synchronization. All of these optimizations hold true for both encoding and decoding.

4 Analysis and Parameter Selection

The performance of our OKVS scheme critically depends on the upper bound $\hat{g}$ of g. We demonstrate that our algorithm is optimal for several important cases and otherwise achieves an extremely good approximation in practice. We then give a detailed characterization of the distribution of $\hat{g}$.

4.1 The Case of $w = 2$

We begin with the case of row weight $w = 2$. This case is qualitatively distinct due to the expected value of g being non-zero for practical values of e and the optimality of our algorithm.

Reformulating as a Graph Problem. In order to analyze the gap g, we reformulate our problem as a graph problem as follows. Recall that the sparse binary part of the matrix H under consideration is of size $n \times m'$ where $m' = en$ for an expansion factor of $e > 1$. We sample the sparse part of H as follows: Each row is sampled independently at random subject to the constraint that the row is of weight $w = 2$. Now, consider the graph $G_H = (V, E)$ induced by the sparse part of H where $V = [m']$ and

$$E = \{e_i = (e_{i,1}, e_{i,2}) : H_{i,1} = H_{i,2} = 1\}_{i \in [n]}$$

It turns out that the problem of determining how to best solve the system of equations determined by H using the process of back-substitution, that is, finding free variables in H, is equivalent to the problem of optimally *peeling* G_H [Wal21]. We recall that peeling G_H corresponds to iteratively selecting an edge with a vertex of degree 1 (or a *free* vertex) and removing that edge. If this process terminates with all the edges of the graph being removed, the graph is said to have been completely peeled. Clearly, in our setting, this means that there is a way to solve the system of linear equations involving H entirely using back-substitution, or in other words, $g = 0$. Otherwise, the peeling algorithm terminates with a sub-graph of G_H where every vertex has degree at least 2. That sub-graph is called the 2-*core* of G_H.

Indeed, different sub-graphs of G_H would admit different peelings that would lead to different such final unpeelable sub-graphs. One can then ask the question of whether one can determine the largest sub-graph of G_H that can be completely peeled. We call this the problem of optimally peeling G_H. Optimally peeling G_H would lead to the identification of a sub-graph (where every vertex has degree at least 2) of minimum size which if removed from G_H would make it completely peelable. It is easy to see that the gap g is the size of this subset, i.e., the minimum number of edges to be removed from this 2-core in order to make it peelable. Immediately, g is upper-bounded by the size of the 2-core of G_H. Thus, in general, one way of upper-bounding g is to get a handle on the size of the 2-core of G_H. But for $w = 2$, we can do a lot better. It turns out that for the case of $w = 2$, it is indeed possible to efficiently compute the optimal peeling of G_H a greedy algorithm. We go on to describe why this is the case.

Matroids and the Greedy Algorithm. We begin by defining the matroid abstraction and cast our algorithm with $w = 2$ as the associated greedy algorithm. This immediately implies that our algorithm is optimal for the case of $w = 2$. A *matroid* $M = (E, I)$ consists of a *ground set* E and a collection $I \subseteq 2^E$ of *independent sets* where each $A \in I$ is a subset of E, i.e., $A \subseteq E$, which satisfies the following three properties:

(1) The empty set is independent, i.e. $\emptyset \in I$.
(2) (**hereditary**) Every subset of an independent set is an independent set, i.e. $A' \subseteq A \in I \implies A' \in I$.
(3) (**augmentation/exchange**) If $A, B \in I$ and $|A| > |B|$, then $\exists a \in A \setminus B$ s.t. $B \cup \{a\} \in I$.

This definition can be extended to the setting where the elements of E each have a positive weight. The weight of any subset $A \subseteq E$ is then defined as the sum of the element weights. Given this, a maximum weight independent set $A \in I$ can be found in $|E|$ time given oracle access to determining set independence [Oxl06], i.e., determining if $A \cup \{a\} \in I$ for some $A \in I, a \in E$. In particular, A can be found by initializing $A := \emptyset$ and greedily adding some $a \in E$ to A so long as $A \cup \{a\} \in I$.

Let us now look back to the problem of optimally peeling G_H, or equivalently, finding the largest sub-graph of G_H that is completely peelable. It is a well-known fact that if $G = (V, E)$ is an undirected graph, and F is the family of sets of edges that form forests in G, then (E, F) forms a matroid. For the sake of completness, we recall the proof of this fact here. Clearly, $\emptyset \in F$. F clearly satisfies the hereditary property as removing edges from a forest leaves another forest. F also satisfies the exchange property: if A and B are both forests, and A has more edges than B, then it has fewer connected components, so by the pigeonhole principle there is a component C of A that contains vertices from two or more components of B. Along any path in C from a vertex in one component of B to a vertex of another component, there must be an edge with endpoints in two components, and this edge may be added to B to produce a forest with more edges. Thus, F forms the independent sets of a matroid, called the *graphic matroid* of G, denoted as $M(G)$.

Let us now consider $M(G_H)$. The independent sets of this matroid are the forests in G_H. Now, note that forests are *completely peelable*, i.e., have empty 2-cores. Thus, finding the largest possible forest embedded in G_H corresponds to finding the optimal way to peel G_H. Taking it full circle, this then also gives us the exact set of rows of H which attains the minimum possible gap g. To this end, we can set the weight of each edge $e_i \in E$ as 1. The greedy algorithm will then determine the maximum weight, i.e., largest size independent set $A \subseteq 2^E$. More concretely, let the ground set $E := [n]$ correspond to the rows of H and we define I such that all $A \in I$ correspond to sets of rows which can be made exactly triangular via row a permutation. From the argument above, clearly (1), (2) and (3) are satisfied. We then define the weight of each $a \in E$ as one. The maximum weight independent set $A \in I$ is then exactly the set of rows which

minimizes g. It is then an easy task to verify that our triangulation algorithm is equivalent to the matroid greedy algorithm and therefore is optimal.

Analysis. From the discussion above, we know that whatever g is, we can efficiently find it. The remaining analysis thus is to bound g with overwhelming probability. Again, we do this via reformulating the problem as a graph problem. The process of sampling the sparse part of H can be equivalently considered as sampling a random graph $G_H = (V, E)$ where $V = [m']$ and $|E| = n$, that is, where E is a collection of random independently (with replacement) sampled n edges. Now, as argued above, the gap g of H is upper-bounded by the size of the 2-core of G_H. Thus, in order to upper bound g, it suffices to understand the distribution of the size of the 2-core of such a randomly sampled G_H.

This has, once again, been studied extensively. As noted in [FK21], the size of the 2-core of G_H follows the size of the so-called *giant component* in G_H. We recall the phenomenon of the emergence of the giant component in a random graph. Asymptotically, the evolution of the random graph undergoes a phase transition at $e = 2$. The case of $e > 2$ is called the *sub-critical* phase and $e < 2$ is called the *super-critical* phase. In the sub-critical phase, the random graph consists with overwhelming probability of tree components and components with exactly one cycle. All components during the sub-critical phase are rather small, of order $\mathcal{O}(\log m')$, and there is no significant gap in the order of the first and the second largest component. The situation changes when $e < 2$, i.e., when we enter the super-critical phase and then with overwhelming probability, the random graph consists of a single giant component (of the order comparable to $\mathcal{O}(m')$), and some number of simple components, i.e., tree components and components with exactly one cycle. One can also observe a clear gap between the order of the largest component (the *giant*) and the second largest component which is of the order $\mathcal{O}(\log m')$. This phenomenon of dramatic change of the typical structure of a random graph is called its phase transition. Erdös and Rényi [ER59], and many others following [Bol84; Luc90] established the phenomenon of the *double-jump*, where as we pass through the phase transition $e = 2$, the size of the largest component changes from $\mathcal{O}(\log m')$ to $m'^{2/3}$ to $\Omega(m')$, and in fact, for $e = 2$, the size of the largest component is $m'^{2/3}$ with overwhelming probability. The size of the 2-core, and the upper bound $\hat{g}$ for the gap of H, grows gradually (unlike the size of k-cores for $k \geq 3$, as noted in [FK21]) and eventually follows the same distribution as that of the largest component of G_H for $e < 2$. Close to $e = 2$, the upper bound $\hat{g}$ on the gap turns out to be $\mathcal{O}(\log m')$, following the size of the largest component when $e > 2$.

The issue of duplicates. One subtle issue that gets overlooked by the above analysis is the occurrence of duplicate rows in H. Indeed, in the context of the induced graph G_H, this would amount to a cycle of length 2, something that would not be considered. However, this event turns out to be the most likely failure scenario in our case and one we analyze separately. The probability that there exists a pair of rows in the sparse part of H that are identical is upper-

bounded by

$$\binom{n}{2}\frac{\binom{m'}{w}\left[\binom{m'}{w}\right]^{n-2}}{\left[\binom{m'}{w}\right]^{n}} = \frac{\binom{n}{2}}{\binom{m'}{w}} \approx \frac{n^2}{2}\cdot\frac{w!}{m'^w} = \mathcal{O}\left(\frac{1}{n^{w-2}}\right)$$

for $w \ll n$. For $w = 2$, this is a constant. Thus, in expectation, we will see a constant number of duplicate rows in H, a case our algorithm will not be able to handle. Indeed, as we increase w, the probability of such duplicates occurring falls significantly. This is yet another reason to consider higher weight $w > 2$.

Concrete Parameters. While the analysis above provides asymptotic guarantees that $\hat{g} = \mathcal{O}(\log n)$ for $e = 2$, the concrete security remains ambiguous. We address this by performing extensive empirical evaluations to determine the constants involved. The general strategy is to run the triangulation algorithm a large number of times on random instances for various n. A gap of g is then given an estimated statistical security of λ bits if $2^{-\lambda}$ fraction of the instances had a gap of at most g. Figure 2 shows the resulting distribution. To ensure statistical accuracy, instances with fewer that 2^4 instances are not plotted, e.g. for $n = 2^{10}$ approximately 2^{27} trials were performed while only $\lambda \leq 23$ is plotted.

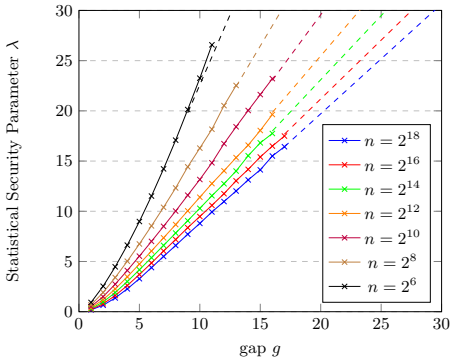

Fig. 2: Security parameter λ for various g and n.

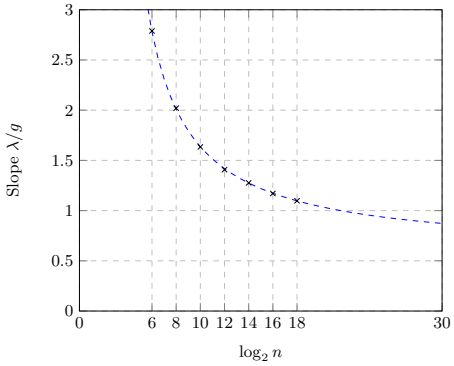

Fig. 3: The slope of Figure 2 trend lines.

We observe two primary relationships:

1. For any fixed n there is generally a linear relationship between λ and g with the exception near zero which is not a practical concern. Since $\lambda = \mathcal{O}(\log n)$ this is consistent with the theoretical analysis.
   For each n we interpolated the best fit line for $g > 5$. Moreover, we observe that all of these linear relationships pass near the point $f = (1.9, 0)$. We conjecture that this is a general trend and plot the best fit line which passes though f as dashed lines.

2. Observe that the slope of the line is λ/g and that we expect this to be linear in $1/\log n$. Indeed, Figure 3 depicts the slopes of the best fit lines of Figure 2 and which follows this relationship. We interpolate this relationship as

$$\lambda/g = a/(\log_2 n - c) + b \tag{1}$$

where $a := 7.529, b = 0.610, c = 2.556$ and is depicted as the dished line.

In summary, we propose a linear lower bound $\lambda \geq \alpha_n g - \alpha_n 1.9$ where $\alpha_n := a/(\log_2 n - c) + b$.

4.2 General Weight

We now consider the case of row weight $w > 2$. As before, in order to analyze the gap g, we reformulate our problem as a graph problem as follows. We sample the sparse part of H as follows: Each row is sampled independently at random subject to the constraint that the row is of weight $w > 2$. Now, consider the hypergraph $G_H = (V, E)$ induced by the sparse part of H where $V = [m']$ and

$$E = \{e_i = (e_{i,1}, \ldots, e_{i,w}) : H_{i,1} = \ldots = H_{i,w} = 1\}_{i \in [n]}$$

Once again, the problem of determining how to best solve the system of equations determined by H using the process of back-substitution, that is, finding free variables in H, is equivalent to the problem of optimally *peeling* G_H, where peeling G_H corresponds to iteratively selecting a hyperedge with a vertex of degree 1 (or a *free* vertex) and removing that hyperedge. If this process terminates with all the hyperedges of the hypergraph being removed, the hypergraph is said to have been completely peeled. Clearly, in our setting, this means that there is a way to solve the system of linear equations involving H entirely using back-substitution, or in other words, $g = 0$. Otherwise, the peeling algorithm terminates with a sub-hypergraph of G_H where every vertex has degree at least 2. That sub-graph is called the 2-*core* of G_H.

Indeed, different sub-graphs of G_H would admit different peelings that would lead to different such final unpeelable sub-graphs. One can then ask the question of whether one can determine the largest sub-graph of G_H that can be completely peeled. We call this the problem of optimally peeling G_H. Optimally peeling G_H would lead to the identification of a sub-graph (where every vertex has degree at least 2) of minimum size which if removed from G_H would make it completely peelable. It is easy to see that the gap g is the size of this subset, i.e., the minimum number of hyperedges to be removed from this 2-core in order to make it peelable. Immediately, g is upper-bounded by the size of the 2-core of G_H. Thus, again, one way of upper-bounding g is to get a handle on the size of the 2-core of G_H. For $w = 2$, it was possible to efficiently compute the optimal peeling of G_H using a greedy algorithm. It turns out that for the case of $w > 2$, it is unlikely that there exists a general procedure to efficiently compute the optimal peeling of G_H. We go on to describe why this is the case.

Hardness of General Weight. We first look into why the matroid-inspired greedy algorithm that worked in the case of $w = 2$ does not work for the case of $w > 2$. To this end, let us attempt defining a matroid for the case of $w > 2$. Recall that in the case of $w = 2$, the matroid corresponding to $G = (V, E)$ was (E, F), where F is the family of sets of edges that form forests in G, which was also exactly the set of sub-graphs whose 2-core was empty. Let us try to do the same for the hypergraph $G_H = (V, E)$. Let us define F to be the set of sub-hypergraphs of G_H with empty 2-core. The question is: Does (E, F) consistitute a matroid? It turns out that the answer is no. While $\emptyset \in F$ and F clearly satisfies the hereditary property (removing hyperedges can only make the 2-core smaller), F does not necessarily satisfy the exchange property. In other words, it is possible that there exist two sub-hypergraphs A and B of G_H which have empty 2-cores, and A has more hyperedges than B, however, adding any hyperedge of A to B induces a non-empty 2-core in B. It is not a difficult exercise to construct examples of this form. More formally, let us define the function $f : 2^E \to \{0, 1\}$ to be the indicator function for non-empty 2-cores, that is, the output of f on a set of hyperedges is 0 if and only if the corresponding sub-hypergraph has an empty 2-core. Defined this way, it is easy to see that f is monotone, that is, $f(S) \leq f(T)$ for all $S \subseteq T$. However f is not necessarily sub-modular or super-modular. Indeed, if $S \subseteq T$ and $e \in E \setminus T$, then we cannot reliably state a relationship between $f(S \cup \{e\}) - f(S)$ and $f(T \cup \{e\}) - f(T)$ (the marginal benefit of adding e to S versus the marginal benefit of adding it to T). Let us consider two cases. First consider the case of S, T, e such that S has an empty 2-core, while $S \cup \{e\}$ and T have non-empty 2-cores. Then, $f(S \cup \{e\}) - f(S) = 1 - 0 = 1$ while $f(T \cup \{e\}) - f(T) = 1 - 1 = 0$. Next, consider the case of S, T, e such that S, T and $S \cup \{e\}$ have an empty 2-cores, while $T \cup \{e\}$ and has a non-empty 2-core. Then, $f(S \cup \{e\}) - f(S) = 0 - 0 = 0$ while $f(T \cup \{e\}) - f(T) = 1 - 0 = 1$. This leads to the observation that our aforementioned greedy algorithm is unlikely to even approximate the optimal peeling of G_H.

Given the above, a next approach would be to consider if defining the independent sets differently would aid in the development of an efficient greedy algorithm to optimally peel G_H. Indeed, a hypercycle matroid over hypergraphs has been considered in the literature [FKK03; For17]. The independent sets in this case are known as hyperforests, but unfortunately, the definition of the hyperforests does allow for them to have non-empty 2-cores.

Finally, one could ask if there is any other efficient algorithm to optimally peel G_H for the case of $w > 2$. This problem is closely connected to the problem of finding minimum unsatisfiable cores in the context of satsifiability. Two works in this regard are those of Zhang, Li and Shen [ZLS06] and Papadimitriou and David Wolfe [PW88]. From the first, we note that even verifying minimum cores is very hard, and that it is known to be D^{P}-complete from the latter ($D^{\mathsf{P}} = \{A \cap B : A \in \mathsf{NP} \wedge B \in \mathsf{coNP}\}$). So this problem is certainly hard and is unlikely to admit efficient algorithms.

Thus, given that determining g optimally might be difficult, and that the running time of our algorithm depends quadratically on g, we are left with two questions:

- How large is g likely to be for the matrices that we sample?
- How does this affect the performance of our algorithm?

Phase Transition. The first observation we make is regarding the phenomenon of a sharp phase transition with regards to the emptiness of the 2-core that occurs with varying the expansion factor e. This phenomenon has been completely described in the work of Dembo and Montanari [DM08]. Recalling (and rephrasing) their main result, for a uniformly chosen random hypergraph of $m' = en$ vertices and n hyperedges, each consisting of the same fixed number $w \geq 3$ of vertices, the size of the 2-core exhibits for large n a first-order phase transition, changing from $o(n)$ for $e > e_c$ to a positive fraction of n for $e < e_c$, with a transition window size $\Theta(n^{-1/2})$ around $e_c > 0$. For instance, for $w = 3$, they estimate $e_c = 1.2218$.

Structures in the Gap. We now know that below a certain critical expansion threshold, the 2-core of G_H becomes non-empty. What does this 2-core of G_H contain? For general w, the minimal larger structures "induce" cycles in the hypergraph $G_H = (V, E)$ induced by the sparse part of H where $V = [m']$ and

$$E = \{e_i = (e_{i,1}, \ldots, e_{i,w}) : H_{i,1} = \ldots = H_{i,w} = 1\}_{i \in [n]}$$

Induced cycles of size 2 correspond to duplicates, which have been analyzed above (and will be discussed ahead). For $k > 2$, the probability that G_H contains an induced cycle of size k is upper-bounded by

$$\binom{n}{k} \frac{\binom{m'}{k} k! \left[\binom{m'-2}{w-2}\right]^k \left[\binom{m'}{w}\right]^{n-k}}{[\binom{m'}{w}]^n} = \frac{\binom{n}{k} \frac{m'!}{(m'-k)!} \left[\binom{m'-2}{w-2}\right]^k}{\left[\binom{m'}{w}\right]^k} < \frac{\binom{n}{k} \frac{m'!}{(m'-k)!} \left[\binom{m'}{w-2}\right]^k}{\left[\binom{m'}{w}\right]^k}$$

This probability is upper-bounded by

$$\lim_{n \to \infty} \frac{\binom{n}{k} \frac{m'!}{(m'-k)!} \left[\binom{m'}{w-2}\right]^k}{\left[\binom{m'}{w}\right]^k} = \frac{\left(\frac{w(w-1)}{c}\right)^k}{k!} \approx \frac{1}{\sqrt{2\pi k}} \left(\frac{\frac{ew(w-1)}{c}}{k}\right)^k = \mathcal{O}\left(\frac{1}{\tilde{k}^{k+\frac{1}{2}}}\right)$$

where $\tilde{k} = \beta(w, c)k$ for some $\beta(w, c) = \Theta(cw^{-2})$.

The above analysis qualitatively states that the minimal structures in the gap are more likely to be small than large. The understanding of this fact will be important as we move ahead.

The issue of duplicates. As before, the issue that gets overlooked by the above analysis is the occurrence of duplicate rows in H. From before, The probability that there exists a pair of rows in the sparse part of H that are identical is $\mathcal{O}(n^{2-w})$ for $w \ll n$.

Putting it all together. Let us now try to describe the what we expect should happen to g as we vary e. Firstly, there is a phase transition window to the far right of which the only bad structures in the 2-core would be duplicates (with a certain probability) and to the far left of which the 2-core would start to contain (almost surely) structures of linear size. Secondly, the width of this phase transition varies as $\Theta(n^{-1/2})$, being wider for smaller n and thinner for larger n. Just to the right of the phase transition window, the 2-core contains, with overwhelming probability, only a few small structures (for instance, duplicates). Through the phase transition, we would expect to see larger structures in the gap as we get closer to the left end of the phase transition window. In other words, towards the right end of the phase transition window, we would still expect that the gap is small with overwhelming probability.

Concrete Parameters. The empirical measurements we make do indeed match with the theoretical analysis above. Figure 4 plots the log failure probability versus the expansion factor $e = m'/n$ for $w = 3$ and various n. For each n we plot several curves. The lower solid curve for each n plots the security parameter for $g < 1$. Some n have a second solid curve which plots the security parameter for $g < 2$. Inspecting the distribution of these curves, we see several trends. The first is that as n increases, the curve increases at a higher rate (in other words, the phase transition window is wider for smaller n).

Secondly, there is a plateauing effect where a given curve starts to flatten out. For example, consider $n = 2^{10}$ and observe that the $g < 1$ curve flattens out at $\lambda \approx 10$ and $e = 1.31$. However, the $g < 2$ curve continues to increase. This plateauing effect perfectly agrees with the probability of small structures. The most likely (virtually only) structures in the gap post the phase transition are duplicate rows which occur with probability approximately $1/n^{w-2} = 1/n$ for $w = 3$. Therefore it is expected to have plateaus at $\lambda = n$ for each $g < 1$ curve. Similarly, the mostly likely structure of size $g < 2$ are two duplicates which, as expected, occurs at $\lambda = 2n$. Figure 4 depicts the projected continuations of these plateaus as the horizontal dashed lines.

The next observation is that, if we ignore the plateau effect, the curves for each n and $g < 1, 2, ...$ converge to a linear growth rate. We project this growth rate for each n as the corresponding linear dashed line. The trend that each of these lines pass thought the point $f = (1.223, -9.2)$, which we call the *phase transition point*. Therefore as n goes to infinity, we would expect a sharp phase transition where g goes from $g > 0$ to $g = 0$ at $e = 1.223$. This closely agrees with [DM08] which analytically computes the phase transition to occur at $e = 1.2218$.

Figure 5 contains a similar plot for $w = 5$. We observe that it generally follows the same trend. The most significant change is that the phase transition point f is shifted right to $f = (1.427, -9.2)$. Secondly, the plateau effect occurs at a higher λ. For example, a duplicate occurs at $\lambda \approx 3\log(n)$ as the probability of duplicate rows for $w = 5$ is $\approx 1/n^{w-2} = 1/n^3$, which can be observed in Figure 5 for $n = 2^6$. Additionally, we performed this analysis for $w \in \{3, 4, 5, 7, 9, 14, 20, 100\}$ and observe that slope of the linear trend lines changes as w is changed.

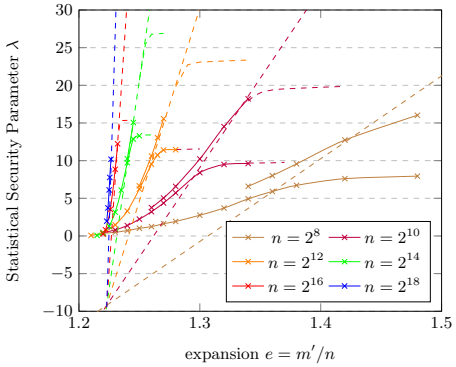

Fig. 4: Security parameter λ for $w = 3$ and various e, n.

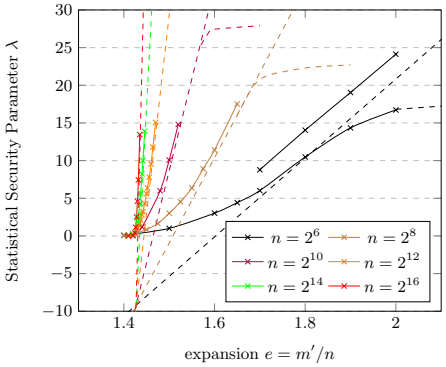

Fig. 5: Security parameter λ for $w = 5$ and various e, n.

Our goal is then to translate these observation into a function of w, e, n which closely lower bounds λ in the range $30 \leq \lambda \leq 128, 3 \leq w \leq 10, 2^6 \leq n \leq 2^{30}$. The general strategy is to upper bound the e coordinate of the phase transition point $f = (e^*, -9.2)$ and lower bound the slope of the linear trend lines. These bounds can then be combined with the well defined impact of small structures and plateaus to obtain a lower bound of λ.

For all $w \in \{3, 4, 5, 7, 9, 14, 20, 100\}$ we observe that the e coordinate of the phase transition point $f = (e^*, -9.2)$ has a positive correlation with w. Figure 6 depicts this relation by plotting e for various w. As can be seen, the relationship is roughly linear. Recall that our objective is derive a function which will lower bound λ given the other parameters, and therefore we need to upper bound e in the curve in Figure 6. We achieve this in a piecewise manner with cases $w = 3, w = 4, w \geq 5$. For the first two cases we simply take the empirically obtained values while for $w \geq 5$ we propose the upper bound

$$e^* = \begin{cases} 1.223 & \text{if } w = 3 \\ 1.293 & \text{if } w = 4 \\ 0.1485w + 0.6845 & \text{otherwise} \end{cases}. \tag{2}$$

This upper bound is depicted as the dashed line in Figure 6. As can be seen at $w = 20$, the empirically obtained values have a slightly sublinear growth which was further validated at $w = 100$. Since we are primarily interested in $w \leq 10$, we argue that this upper bound is sufficiently accurate.

Finally, we turn our attention to understanding the distribution of the slope (λ/e) of the linear trend lines. Figure 7 shows this relationship between the log slope and log n for various w. It is immediately clear that there is a linear relationship between $\log(\lambda/e)$ and $\log(n)$. The slope of this relationship is 0.55. This closely agrees with the predicted value of $1/2$ from [DM08]. Indeed, [DM08] states that the width of the phase transition varies as $\Theta(n^{-1/2})$. The slope of the phase transition line is inversely related to the width of the phase transition

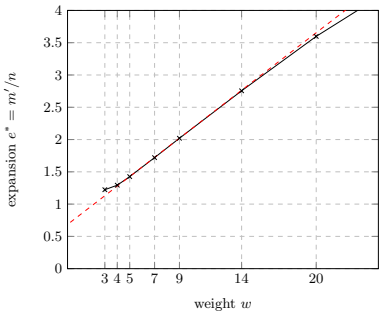

Fig. 6: phase transition point e-coordinate for various w

window, that is, $\lambda/e \propto \Theta(n^{1/2})$. Thus, theoretically, the slope of the linear relationship between $\log(\lambda/e)$ and $\log(n)$ would be $1/2$, which closely matches our empirically obtained value of 0.55.

As expected, the slope is the same for all measured w but each line has has a different additive offset. To understand this offset we plot the same measurements in Figure 8 where we now compare the log slope with log w for various n. Here the linear growth rate in Figure 7 corresponds to even spacing between all the curves. We observe that $w \in \{3, 4, 5\}$ all have approximately the same slope with $w = 4$ being slightly higher. Then as w increases we observe a general trend of the slopes decreasing. Again, our goal is to lower bound the slope. We achieve this via the polynomial

$$\log_2(\lambda/e) = 0.555 \log_2(n) + 0.0930 w'^3 - 1.010 w'^2 + 2.925 w' - 0.133 \quad (3)$$

for the range $3 \leq w \leq 20$ and $w' := \log_2(w)$. We note that outside this range the polynomial quickly becomes an upper bound and further investigation is required to obtain a more general lower bound. We plot this lower bound in Figure 8 as the dished lines. In summary, one can solve Equation 2 and 3 to obtain the phase transition line. It is then a simple matter to compute e and $\hat{g} = \lfloor \frac{\lambda}{(w-2)\log_2(en)} \rfloor$.

4.3 Performance of our algorithm

Optimal for $g = 0$. Our first positive result for general w is that our algorithm is optimal for the case of $g = 0$. Revisiting step 2 of the algorithm, note that for $g = 0$, $\delta = n$. This means that the rows can be ordered such that every row of H has a corresponding sparse column which contains a non-zero entry in that row and some of the subsequent rows. This corresponds exactly with the case of the hypergraph induced by the sparse part of H having an empty 2-core and hence can be completely peeled. Indeed, in this case, the optimal way of solving the system is via back-propagation which is done in $\mathcal{O}(m)$ time by step 5 of our algorithm.

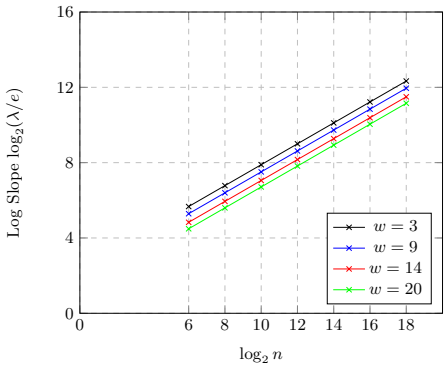

Fig. 7: Slope of the phase transition λ/e for various n.

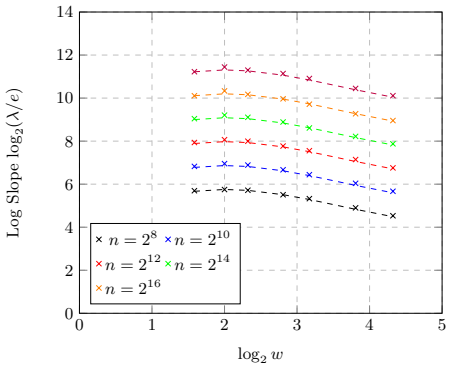

Fig. 8: Slope of the phase transition λ/e for various w

We would like to argue that if H had a triangulation that enabled $g = 0$, step 2 of our algorithm indeed finds one such triangulation. This is in fact very easy to see. If H admitted such a triangulation, then there must exist a column of H that has a single non-zero entry. Thus, the first iteration of our algorithm would pick out such a column. Let us now complete the argument inductively. Consider H' after one row of H has been removed, that is, a row with a column whose only non-zero entry lies in that row. Note that H is triangularizable with $g = 0$ iff H' is. Thus, to complete the inductive argument, our algorithm optimally triangulates H' with $g = 0$, and then simply re-introducing the removed row preserves the triangulation as the row has a corresponding column whose only non-zero entry lies in that row. Alternatively, we can view this via the lens of optimally peeling G_H, i.e., step 2 of our algorithm completely peels G_H if it can be peeled.

Optimal for duplicates. We note that our algorithm is optimal for the case that the only structures in the 2-core are duplicate rows. In this case, it is easy to see that step 2 of our algorithm will triangulate H but for its duplicate rows. Or equivalently, our algorithm will peel G_H except for the identical hyperedges.

The case of non-duplicates. If the structures in the 2-core include more than just duplicate rows, it is hard to theoretically bound the quality of the g determined by our algorithm. Indeed, our algorithm does perform better than prior works as the upper bound $\hat{g}$ on the gap g as determined by our algorithm is strictly smaller than the size of the 2-core of G_H. However, in the worst case, $\hat{g}$ could end up being as large as the size of the 2-core (minus one). While this does seem less than optimal, it is unclear if one can do significantly better (efficiently in terms of g) than this on account of the hardness results presented in Section 4.2. We discuss this in more detail ahead.

Probabilistically good approximation. The silver lining is that probabilistically, our algorithm is guaranteed to perform really well. Our analysis from before has left us with the understanding that the probability that the 2-core of G_H contains large structures declines (slightly faster than) exponentially with their size. Therefore, we are guaranteed that with high probability, the structures in the 2-core are indeed small and few in number. Furthermore, as empirically verified above, we observe that duplicates and small cycles (contributing a gap of 1 or 2) are the most likely structures in the 2-core of G_H and our algorithm works optimally in the case of just duplicates. Although the asymptotic analysis is not tight, it does provide enough theoretical evidence that g will be small (e.g. < 12), and empirically, we note that g is even smaller (e.g. 2 or 3).

Finally, the above analysis also heuristically guarantees that our algorithm, with overwhelming probability, will not veer off significantly from the actual gap. The reason for this is the following. Assume that our algorithm has to make a choice at some stage in step 2 on which column to pick among many of equal minimum non-zero weight. In the case that it makes a "wrong" choice that results in a large gap, this would mean that the sub-matrix H' (where remaining rows follow the same distribution as only the column weight has been affected by removing rows in step 2) would have to contain a sufficient number of large-enough independent structures, which, again, as argued above, is unlikely. This provides theoretical evidence for our empirical finding that our algorithm indeed performs near-optimally.

Optimality for small g. We finally discuss the optmial triangulation described in Figure 12. Optimally peeling G_H amounts to identifying a minimum subset of hyperedges of the 2-core of G_H which when removed make G_H completely peelable. Indeed, if we had an upper bound $\hat{g}$ on the size of that subset of the 2-core, an exhaustive search among the edges of the 2-core would reveal the optimum (minimum) subset of hyperedges (which directly maps to the gap of H). Given the hardness results presented in Section 4.2, an exhaustive search is probably the best one could hope for, and this is exactly what is done in Figure 12. The first steps end with H' containing those rows corresponding to the 2-core of G_H. Subsequently we perform an exhaustive search to determine the optimum subset of rows for the gap. Let h_2 denote the size of the 2-core of G_H. Then, the complexity of the modified triangulation step would be $\mathcal{O}(m + h_2^{\hat{g}})$. From all of the analysis above, indeed in the parameter regime we consider, $\hat{g} = \mathcal{O}(1)$ and $h_2 \leq c \cdot \hat{g}$ with overwhelming probability for $c = \mathcal{O}(1)$. In fact, $c \approx 2$ with overwhelming probability and $\hat{g}$ is a small constant (for instance, between 2 and 5). Thus, in our setting, it is indeed possible to optimally triangulate H (equivalently, optimally peel G_H) efficiently (in linear time).

5 Subfield VOLE and PSI

Next we turn to the main application for our construction, efficient Private Set Intersection. PSI allows two parties to compute the intersection of their

respective sets X, Y without revealing anything but the final intersection. The full functionality is detailed in Figure 9. To realize this functionality we make use of the recent PSI construction of [RS21] and modify it in two ways.

1. Replace the OKVS of [PRTY20] with our new construction.
2. Optimize the PSI construction of [RS21] to use subfield-VOLE [CRR21].

As detailed below, the former alteration improves both the computational overhead and communication overhead. In particular, when we instantiate the [RS21] PSI protocol with our new OKVS, the PSI protocol is able to be executed on a consumer laptop in 0.4 seconds on a single thread, or 0.19 seconds on multiple threads. Additionally, when the subfield-VOLE optimization is applied we are able to further reduce the communication overhead.

Parameters: There are two parties, a sender with set $Y \subset \mathbb{F}$ and a receiver with a set of key $X \subseteq \mathbb{F}$. Let $n_{\mathsf{y}}, n_{\mathsf{x}}, n_{\mathsf{x}}' \in \mathbb{Z}$ be public parameters where $n_{\mathsf{x}} \leq n_{\mathsf{x}}'$.

Functionality: Upon receiving $(\mathsf{sender}, \mathsf{sid}, Y)$ from the sender and $(\mathsf{receiver}, \mathsf{sid}, X)$ from the receiver. If $|Y| > n_{\mathsf{y}}$, abort. If the receiver is malicious and $|X| > n_{\mathsf{x}}$, then abort. If the receiver is honest and $|X| > n_{\mathsf{x}}$, then abort.
The functionality outputs $X \cap Y$ to the receiver.

Fig. 9: Ideal functionality $\mathcal{F}_{\mathsf{psi}}$ of Private Set Intersection [RS21].

Parameters: There are two parties, a Sender and a Receiver. Let $\mathbb{F}$ be an extension field over base field $\mathbb{B}$. Let m denote the size of the output vectors.

Functionality: Upon receiving $(\mathsf{sender}, \mathsf{sid})$ from the Sender and $(\mathsf{receiver}, \mathsf{sid})$ from the Receiver. Sample $\vec{A} \leftarrow \mathbb{B}^m, \vec{B} \leftarrow \mathbb{F}^m, \Delta \leftarrow \mathbb{F}$ and compute $\vec{C} := \vec{A}\Delta + \vec{B}$. The functionality sends $\Delta, \vec{B}$ to the Sender and $\vec{C} := \vec{A}\Delta + \vec{B}, \vec{A}$ to the Receiver.

Fig. 10: Ideal functionality $\mathcal{F}_{\mathsf{sub\text{-}vole}}$ of semi-honest random Sub-field Vector-OLE.

Let us review the necessary building blocks. We make use of a subfield random Vector Oblivious Function Evaluation (VOLE). Our starting point is the VOLE protocol of [CRR21] which itself is a modification of [BCG+19]. Although [CRR21] does not explicitly mention it, their construction naturally supports subfield VOLE as described by [BCG+19]. The VOLE functionality described in Figure 10 consists of a Sender and Receiver. The Sender is output a random $\Delta \in \mathbb{F}, \vec{B} \in \mathbb{F}^m$ while the Receiver is output random vectors $\vec{A} \in \mathbb{B}^m, \vec{C} \in \mathbb{F}^m$ subject to

$$\vec{C} - \vec{B} = \Delta \vec{A}$$

where $\mathbb{F}$ is some extension field of $\mathbb{B}$. For example, $\mathbb{B} = \mathbb{F}_p$ while $\mathbb{F} = \mathbb{F}_{p^d}$ for some prime p and extension degree d. [RS21] presented a PSI protocol given a VOLE correlation. The PSI functionality is presented in Figure 9. To obtain PSI, the intuitive idea is to derandomize the VOLE correlation such that we obtain $\vec{C}' - \vec{B}' = \Delta\vec{P}$ where $\vec{P}$ is a OKVS which decodes to $\mathsf{H}^{\mathbb{B}}(x)$ for all $x \in X$, where $\mathsf{H}^{\mathbb{B}} : \{0,1\}^* \to \mathbb{B}$ is a random oracle. Then, by the linearity of the OKVS scheme and for $x \in X$, we have

$$\mathsf{Decode}(\vec{C}', x) - \mathsf{Decode}(\vec{B}, x) = \Delta\mathsf{Decode}(\vec{P}, x) = \Delta\mathsf{H}^{\mathbb{B}}(x)$$
$$\mathsf{Decode}(\vec{C}', x) - \Delta\mathsf{H}^{\mathbb{B}}(x) = \mathsf{Decode}(\vec{B}', x)$$

The Sender with $\vec{C}, \Delta$ is able to compute the left hand side while the Receiver with $\vec{B}$ is able to compute the right hand side. The Receiver computes $X' := \{F(x) \mid x \in X\}$ where $F(x) := \mathsf{H}^{\circ}(\mathsf{Decode}(\vec{B}', x))$, while the Sender can compute $F(y) = \mathsf{H}^{\circ}(\mathsf{Decode}(\vec{C}', y) - \Delta\mathsf{H}^{\mathbb{B}}(y))$ for any y. The PSI protocol completes by having the Sender send $Y' := \{F(y) \mid y \in Y\}$ to the Receiver who can then infer the intersection $X \cap Y$ from $X' \cap Y'$. The full protocol is presented in Figure 11.

Compared to the protocol of [RS21], we have made several alterations which optimize it for the semi-honest setting along with the use of subfield VOLE. In particular, [RS21] is presented in the malicious setting and utilizes the abstraction of an OPRF. Due to technical issues in the proof, [RS21] requires some additional randomization which can be omitted in the semi-honest setting, i.e. $w^{\mathsf{s}}, w^{\mathsf{r}}, w$ and needing to input x, y to H°. More importantly, we generalize the construction to allow the use of subfield VOLE. When $\mathbb{B} = \mathbb{F}$ we effectively obtain a semi-honest version of [RS21]. However, we consider the case of $\mathbb{F}$ being an extension field of $\mathbb{B}$. Next we demonstrate that our new protocol is secure given that $\log_2 |\mathbb{B}| \geq \lambda + \log_2(n_{\mathsf{x}}) + \log_2(n_{\mathsf{y}})$ and $\log_2 |\mathbb{F}| \geq \kappa$.

Theorem 1. *The Protocol Π_{psi} realizes the $\mathcal{F}_{\mathsf{psi}}$ functionality against a semi-honest adversary in the $\mathcal{F}_{\mathsf{sub\text{-}vole}}$-hybrid model.*

The proof of Theorem 1 follows the same structure as that of [RS21]. Here we outline the most important changes. With respect to correctness, there are effectively three failure modes of the construction.

1. First is that our encoder may fail to solve the system in step 1. However, this occurs with negligible probability.
2. The second failure mode is that for some $y \in Y \setminus X, x \in X$, there is a collision $\mathsf{H}^{\mathbb{B}}(y, r) = \mathsf{H}^{\mathbb{B}}(x, r)$. Since X, Y are first fixed and then r is uniformly sampled, the probability of a collision is at most the probability two random subsets of $\mathbb{B}$ of size $n_{\mathsf{x}}, n_{\mathsf{y}}$ having a non-empty intersection. Taking the union bound over the $n_{\mathsf{x}} n_{\mathsf{y}}$ pairs we obtained $\Pr[\text{collision}] \leq \frac{n_{\mathsf{x}} n_{\mathsf{y}}}{|\mathbb{B}|}$ which is at most $2^{-\lambda}$ when $|\mathbb{B}| \geq 2^{\lambda} n_{\mathsf{x}} n_{\mathsf{y}}$.
3. The final failure mode is if there is a spurious collision between X', Y' given distinct inputs to H°. This effectively has the same failure probability as the previous case given that $\mathsf{out} = \lambda + \log_2(n_{\mathsf{x}}) + \log_2(n_{\mathsf{y}})$.

With respect to privacy, the core task is to demonstrate that Y' can be simulated given only the intersection. The Receiver can distinguish $y' \in Y' \setminus X'$ from random if it so happens that $\mathsf{Decode}(\vec{P}, y') = \mathsf{H}^{\mathbb{B}}(y', r)$. Observe that $\vec{P}$ is a function of X, r and is therefore independent of $\mathsf{H}^{\mathbb{B}}(y', r)$ given that $y' \notin X$ and $\mathsf{H}^{\mathbb{B}}$ is a random oracle. Therefore, the Receiver can distinguish with probability at most $\frac{n_{\mathsf{y}}}{|\mathbb{B}|} \leq 2^{-\lambda}$ given that $|\mathbb{B}| \geq 2^{\lambda} n_{\mathsf{y}} n_{\mathsf{x}}$.

Small domain optimizations. The construction can be further optimized given that the input elements can be expressed in $\lambda + \log_2(n_{\mathsf{y}})$ bits. The core idea is that we can define $\mathsf{H}^{\mathbb{B}}{}_r(x) := \mathsf{H}^{\mathbb{B}}(x, r)$ as a random permutation. Given this, observe that the probability of a $\mathsf{H}^{\mathbb{B}}(y, r) = \mathsf{H}^{\mathbb{B}}(x, r)$ collision for $y \neq x$ is zero. The second impact this change has is to the simulatibility of the sender's final message. Recall that the receiver can distinguish with probability at most $\frac{n_{\mathsf{y}}}{|\mathbb{B}|}$ and therefore, the protocol is secure given that $\log_2 |\mathbb{B}| \geq \lambda + \log_2(n_{\mathsf{y}})$.

Parameters: There are two parties, a Sender and a Receiver with a respective set Y, X where $|Y| = n_{\mathsf{y}}, |X| = n_{\mathsf{x}}$. Let $\mathbb{B}$ be a field with extension $\mathbb{F}$ such that $|\mathbb{F}| = O(2^{\kappa})$. Let $\mathsf{H}^{\mathbb{B}} : \{0,1\}^* \to \mathbb{B}, \mathsf{H}^{\mathsf{o}} : \{0,1\}^* \to \{0,1\}^{\mathsf{out}}$ be random oracles.

Protocol: Upon input $(\mathsf{sender}, \mathsf{sid}, Y)$ from the Sender and $(\mathsf{receiver}, \mathsf{sid}, X)$ from the Receiver, the protocol is specified as:

1. The Receiver samples $r \leftarrow \{0,1\}^{\kappa}$ and computes $\vec{P} := \mathsf{Encode}(L, r)$ where $L := \{(x, \mathsf{H}^{\mathbb{B}}(x, r)) \mid x \in X\}$.
2. The Sender sends $(\mathsf{sender}, \mathsf{sid})$ and the Receiver sends $(\mathsf{receiver}, \mathsf{sid})$ to $\mathcal{F}_{\mathsf{sub\text{-}vole}}$ with dimension $m := |\vec{P}|$, base field $\mathbb{B}$ and extension $\mathbb{F}$. The parties respectively receive $\Delta, \vec{B}$ and $\vec{C} := \vec{A}'\Delta + \vec{B}, \vec{A}'$.
3. The Receiver sends $r, \vec{A} := \vec{P} + \vec{A}' \in \mathbb{B}^m$ to the Sender who defines $\vec{K} := \vec{B} + \vec{A}\Delta \in \mathbb{F}^m$.
4. The Sender sends $Y' := \{\mathsf{H}^{\mathsf{o}}(\mathsf{Decode}(\vec{K}, y, r) - \Delta \mathsf{H}^{\mathbb{B}}(y)) \mid y \in Y\}$ to the Receiver.
5. The Receiver outputs $\{x \in X \mid \mathsf{H}^{\mathsf{o}}(\mathsf{Decode}(\vec{C}, x)) \in Y'\}$.

Fig. 11: Semi-honest protocol Π_{psi} which realizes the PSI functionality $\mathcal{F}_{\mathsf{psi}}$.

6 Evaluation

We now turn our attention to the concrete performance of our constructions and a comparison to related work. In summary, our constructions outperform all prior works by a significant margin. The protocols were bench-marked on a single laptop with Intel i7 9750H and 16GB of RAM. All constructions target $\lambda = 40$ bits of statistical security and $\kappa = 128$ bits of computational security. All protocols were run in the LAN setting with sub millisecond latency. MT denotes that 4 threads per party are used.

6.1 OKVS

We begin by comparing the running time of the encoding and decoding of our constructions and that of [GPR+21]. As shown in Figure 1, our ($w = 3$) threaded constitution requires 653 milliseconds to encode 2^{20} items and just 54 milliseconds to decode them. This is 13× and 64× faster than the ($w = 3$) construction of [GPR+21], respectively. We attribute this speedup to several factors: smaller expansion, approximately 20× fewer dense columns, our faster encoding algorithm and likely a more optimized implementation. Our construction which uses clustering with size $\beta = 2^{14}$ requires just 143 milliseconds compared to 4,912 for the star construction of [GPR+21], a speedup of 34×. We attribute this additional speedup, as compared to $w = 3$, to their star construction requiring a two phase encoding in order to amplify the success probability to be at least $1 - 2^{-\lambda}$. Moreover, their construction & parameters are not optimized for memory efficiency. In addition, our basic $w = 3$ construction is fully secure and therefore our construction allows encoding to be performed in a single pass, with independence between the clusters. This independence property naturally enables multi-threading which can provide an additional 3× speedup with 4 threads.

We also consider the running time of our construction with $w \in \{2, 5\}$ and observe that $w = 3$ outperforms them in terms of running time and compactness. As such we conclude that $w = 3$ is the correct choice for most applications. However, we note that $w > 3$ can have advantages in that larger w can allow for the complete removal of all dense columns. Therefore, if the overall weight needs to be minimized, then $w = 5$ or larger may be preferable.

Looking more closely at the [GPR+21] construction, we observe several differences and noteworthy details. First is that the encoder of [GPR+21] follows a different strategy: 1) greedily traverse the graph to identify the 2-core, 2) solve the *whole* subsystem formed by the 2-core using a quadratic time algorithm, 3) greedily traverse the graph to solve the remaining rows. Their construction leads to several disadvantages as compared to ours. First is the fact that they directly solve the subsystem formed by the 2-core, for which they require $\lambda + |\text{2-core}|$ additional dense columns. This is in contrast to our construction which identifies an *optimal* subset of the 2-core such that remaining system is solvable in linear time, which requires only $|\text{subset}|$ dense columns. For example, with $n = 2^{20}$ our construction requires two dense columns as compared to the estimated value of 60 that [GPR+21] proposed.

Secondly, their encoder requires two graph traversals. The first is needed to identify the 2-core and the second is needed to identify in which order the non-2-core variables should be solved in. While being linear time, this type of graph traversal is a costly operation. In contrast, our construction allows us to maintain an ordered list which exactly determines the order in which the rows/columns should be solved, thereby reduce the running time.

Moreover, [GPR+21] leaves the security guarantees of their $w = 3$ construction largely unclear. In particular, they empirically measure and report the failure probability for the parameters $n = 6600, e = 1.3$ and report that out of 2^{33} trials only a single instance had a 2-core with more than $0.5 \log n$ rows. They

then conclude that with a confidence level of 0.9999, this $0.5 \log n$ upper bound on the 2-core holds with probability at least $1 - 2^{-29}$. However, the probability of this is extremely low since one can show analytically that more than this many small structures are expected to occur. Both their paper [GPR+21] and implementation [Obd] use this upper bound which appears to be in error. However, by our calculation, this oversight likely only reduces their statistical security by a few bits, e.g. $\lambda = 27$ as opposed to 29. The situation is further complicated by the fact that [GPR+21] suggesting that an expansion of $e = 1.3$ generally results in 40 bits of security and go on to use this $e = 1.3$ for both smaller and larger values of n, see [GPR+21, Figure 4, Table 1]. Regardless, this confusion speaks to the necessity of performing a rigorous analysis of the failure probability as has been done in this work.

To obtain provable security with an arbitrarily small failure probability, [GPR+21] proposes a clever amplification technique which takes a OKVS scheme for n' items with failure probability p and amplifies it to p^c for some small c. Moreover, the new amplified construction allows for $n \geq n'$ items to be encoded. This effectively allows them to achieve the standard $\lambda = 40$ bits of security given their $w = 3$ construction. Unfortunately, the amplification roughly requires that each item be encoded/decoded twice. In contrast, our $w = 3$ construction can be tuned to achieve the desired security level directly with little overhead. We refer to this amplified scheme as "$w = 3$, star". The overall expansion rate of this construction is $e = 1.32$ at $n = 2^{20}$. In contrast, our construction requires $e = 1.23$ without clustering and $e = 1.28$ with clustering.

Table 1: The running time (ms) of encoding and decoding various constructions and input sizes n. The elements are $\mathbb{F}_{2^{128}}$ for our construction and $\mathbb{F}_{2^{64}}$ for [GPR+21]. Binary r̂ow denotes that the dense columns are binary and opposed to the default of field r̂ow, see Section 2.

Construction	Encode (ms)			Decode (ms)		
	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}
Ours ($w = 2$)	22.4	878	14,729	7.1	165	3,390
Ours ($w = 3$)	12.1	653	15,362	2.0	54	1,394
Ours ($w = 3$, binary r̂ow)	13.4	680	15,832	3.6	82	1,922
Ours ($w = 5$)	20	1251	30,206	6.4	144	3,317
Ours ($w = 3$, cluster)	**9.5**	**143**	**2,412**	**1.7**	**42**	**930**
Ours ($w = 3$, cluster, MT)	**6.3**	**47**	**764**	**1.1**	**14**	**379**
[GPR+21] ($w = 3$)	359	8,420	158,165	195	3,484	61,014
[GPR+21] ($w = 3$, star)	419	4,912	–	282	5,475	–

6.2 Private Set Intersection

As discussed our main application is to apply our improved OKVS scheme to the PSI protocol of [RS21]. We make several optimizations to their protocol. First is

Table 2: Performance metrics of our PSI protocols compared to related works.

Protocol	Time (ms)			Comm. (bits/n)			Comm. asymptotic (bits)
	2^{16}	2^{20}	2^{24}	2^{16}	2^{20}	2^{24}	n_x, n_y
Semi-Honest							
[KKRT16]	137	2,073	53,933	$984n$	$1008n$	$1032n$	$6\kappa n_x + 3(\lambda + \log(n_x n_y))n_y$
[PRTY20] ($w = 2$)	763	4,998	123,800	$1208n$	$1268n$	$1302n$	$9.3\kappa n_x + (\lambda + \log(n_x n_y))n_y$
[GPR+21] ($w = 3$, star)	180	2,268	–	$780n$	$788n$	$804n$	$5.6\kappa n_x + (\lambda + \log(n_x n_y))n_y$
[RS21] ($w = 2$)	4,995	45,802	113,994	$914n$	$426n$	$398n$	$2^{24} {n_x}^{0.05} + 307 n_x + 40 n_y + \log(n_x n_y) n_y$
Ours (fast)	**51**	**369**	**6,987**	$241n$	$251n$	$260n$	$1.3\kappa n_x + (\lambda + \log(n_x n_y))n_y + 2^{14.5}\kappa$
Ours (fast, MT)	**49**	**163**	**3,145**				
Ours (low comm.)	63	1,353	27,681	**187n**	**157n**	**167n**	$1.2 \log(n_y) n_x + (\lambda + \log(n_x n_y))n_y + 2^{14.5}\kappa$
Ours (low comm., MT)	61	1,107	25,325				
Malicious							
[PRTY20] ($w = 2$)	769	5,196	126,294	$1766n$			$11.8\kappa n_x + 2\kappa n_y$
[GPR+21]($w = 3$, star)	184	2,291	–	$1357n$			$8.6\kappa n_x + 2\kappa n_y$
[RS21] ($w = 2$)	556	5,228	132,951	$960n$	$474n$	$438n$	$2.4\kappa n_x + \kappa n_y + 2^{17} \kappa {n_x}^{0.05}$
Ours	**62**	**439**	**8,055**	**343n**	**302n**	**300n**	$1.3\kappa n_x + \kappa n_y + 2^{14.5}\kappa$
Ours (MT)	65	**222**	**3,984**				

that we replace the OKVS scheme of [PRTY20] with our new constructions. In the semi-honest setting we employ two different schemes. The first is our "fast" variant which using $w = 3$ and a cluster of size 2^{14}. For $n = 2^{20}$ the resulting expansion factor is $e \approx 1.28$. The second version, dubbed "low comm.", uses $w = 3$ without clustering along with the subfield optimization of Section 5. This construction uses $e \approx 1.23$ for $n = 2^{20}$ with $\lambda + \log_2(n_y)$ bit elements, i.e. we employ the small domain optimization of Section 5. In the malicious setting we only consider the "fast" variant. Finally, both of our variants make use of the improved VOLE construction of [CRR21].

In all cases our protocols significantly outperforms the competition. The previous fastest semi-honest protocol was that of [KKRT16] which uses cuckoo hashing and a specialized OPRF-type protocol. Our protocols are between 3 to 8 times faster on a single thread. Our protocol also sends approximately 2 to 5 times less data. Moreover, our fast protocol (with clustering) naturally enables a multi threaded implementation, which brings the overall running time to 0.16 seconds for $n = 2^{20}$, a speedup of 13× compared to [KKRT16]. Compared to [RS21] on which our protocol is based, at $n = 2^{20}$ we observe a speedup of 125× for our protocol and 40× for our low communication protocol while reducing the communication overhead 1.7× and 2.2×, respectively.

Our protocols also outperform [GPR+21] in running time and communication by several fold. [GPR+21] is based on the PSI protocol of [PRTY20] with the original $w = 2$ OKVS scheme being replaced with $w = 3$, star scheme discussed above. Finally, in the malicious setting we observe that our protocol has almost no additional overhead. As such, we similarly outperform all prior works by several fold in running time and communication.

References

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 291–308. ACM, 2019.

[BKM+20] Prasad Buddhavarapu, Andrew Knox, Payman Mohassel, Shubho Sengupta, Erik Taubeneck, and Vlad Vlaskin. Private matching for compute. *IACR Cryptol. ePrint Arch.*, 2020:599, 2020.

[Bol84] Béla Bollobás. The evolution of random graphs. *Transactions of the American Mathematical Society*, 286(1):257–274, 1984.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534. Springer, 2021.

[CT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *Financial Cryptography*, volume 6052 of *Lecture Notes in Computer Science*, pages 143–159. Springer, 2010.

[DM08] Amir Dembo and Andrea Montanari. Finite size scaling for the core of large random hypergraphs. *The Annals of Applied Probability*, 18(5), 2008.

[ER59] Paul Erdös and Alfréd Rényi. On random graphs i. *Publ. Math. Debrecen*, 6:290–397, 1959.

[FK21] Alan Frieze and Michal Karoński. Introduction to random graphs, 2021.

[FKK03] András Frank, Tamás Király, and Matthias Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discret. Appl. Math.*, 131(2):373–383, 2003.

[For17] Quentin Fortier. Aspects of connectivity with matroid constraints in graphs modeling and simulation. *Université Grenoble Alpes*, NNT : 2017GREAM059, 2017.

[GPR+21] 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, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 395–425. Springer, 2021.

[HEK12] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012. URL: https://www.ndss-symposium.org/ndss2012/private-set-intersection-are-garbled-circuits-better-custom-protocols.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

[IKN+20] 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 *EuroS&P*, pages 370–389. IEEE, 2020.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. *IACR Cryptol. ePrint Arch.*:799, 2016. URL: http://eprint.iacr.org/2016/799.

[Luc90] Tomasz Luczak. Component behavior near the critical point of the random graph process. *Random Struct. Algorithms*, 1(3):287–310, 1990.

[Mea86] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, pages 134–137. IEEE Computer Society, 1986.

[NTY21] Ofri Nevo, Ni Trieu, and Avishay Yanai. Simple, fast malicious multiparty private set intersection, 2021. https://ia.cr/2021/1221.

[Obd] OBDBasedPSI. github.com/cryptobiu/OBDBasedPSI.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-n OT extension with application to private set intersection. In *CT-RSA*, volume 10159 of *Lecture Notes in Computer Science*, pages 381–396. Springer, 2017.

[Oxl06] J.G. Oxley. *Matroid Theory*. Oxford graduate texts in mathematics. Oxford University Press, 2006. ISBN: 9780199202508. URL: https://books.google.com/books?id=puKta1Hdz-8C.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spotlight: lightweight private set intersection from sparse OT extension. In *CRYPTO (3)*, volume 11694 of *Lecture Notes in Computer Science*, pages 401–431. Springer, 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from paxos: fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia,*

May 10-14, 2020, Proceedings, Part II, volume 12106 of *Lecture Notes in Computer Science*, pages 739–767. Springer, 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: private set intersection using permutation-based hashing. In *USENIX Security Symposium*, pages 515–530. USENIX Association, 2015.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *USENIX Security Symposium*, pages 797–812. USENIX Association, 2014.

[PW88] Christos H. Papadimitriou and David Wolfe. The complexity of facets resolved. *J. Comput. Syst. Sci.*, 37(1):2–13, 1988.

[RR17] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In *ACM Conference on Computer and Communications Security*, pages 1229–1242. ACM, 2017.

[RS21] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 901–930. Springer, 2021.

[Wal21] Stefan Walzer. Peeling close to the orientability threshold - spatial coupling in hashing-based data structures. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2194–2211. SIAM, 2021.

[ZLS06] Jianmin Zhang, Sikun Li, and ShengYu Shen. Extracting minimum unsatisfiable cores with a greedy genetic algorithm. In Abdul Sattar and Byeong-Ho Kang, editors, *AI 2006: Advances in Artificial Intelligence, 19th Australian Joint Conference on Artificial Intelligence, Hobart, Australia, December 4-8, 2006, Proceedings*, volume 4304 of *Lecture Notes in Computer Science*, pages 847–856. Springer, 2006.

SUPPLEMENTARY MATERIAL

Parameters (apart from Figure 1): Upper bound $\hat{g}$ on the size of the gap.

Encode $((z_1, v_1), ..., (z_n, v_n); r)$:

1. ...
2. **[Opt-Triangulate]** Let $H' := H, J := \emptyset$. While H' has rows:
   (a) Select $j \in [m]$ such that the jth (sparse) column of H' has weight 1.
   (b) Append index j to the ordered list J. Remove the row $i \in [n]$ from H' for which $H'_{i,j} \neq 0$.

   Define $\mathbb{X}$ to be the set of all subsets of size $\leq \hat{g}$ of the remaining rows of H'. For each $X \in \mathbb{X}$, define $H'_X := H' \setminus X$, that is, H'_X is the sub-matrix of H' without the rows indexed by X. Also define the ordered list $J_X := J$. While H'_X has rows:
   (a) Select $j \in [m]$ such that the jth (sparse) column of H'_X has weight 1.
   (b) Append index j to the ordered list J_X. Remove the row $i \in [n]$ from H'_X for which $H'_{X,i,j} \neq 0$.

   Let X^* be an X of minimum size for which the above process terminates with the removal of all rows of H'_X. Define $\delta := |J_{X^*}|$, the gap as $g := n - \delta$, permutation matrices $\pi^{\mathsf{r}} \in \{0,1\}^{n \times n}, \pi^{\mathsf{c}} \in \{0,1\}^{m \times m}$ such that ${\pi^{\mathsf{c}}}_{m-k, m-\delta-k} = 1$ for $k \in [0, \hat{m})$, ${\pi^{\mathsf{c}}}_{J_{X^*,i}, m+1-i} = 1$ and ${\pi^{\mathsf{r}}}_{n+1-i, i'} = 1$ for some i' where $H_{i', J_{X^*,i}} \neq 0$ and all $i \in [\delta]$. Let

$$T := \pi^{\mathsf{r}} \cdot H \cdot \pi^{\mathsf{c}} = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix}.$$

   where $F \in \{0,1\}^{\delta \times \delta}$ is lower triangular, $B \in \mathbb{F}^{g \times \hat{m}}, E \in \mathbb{F}^{\delta \times \hat{m}}$ are the dense columns.
3. ...
4. ...
5. ...

Decode $(\vec{P}, z, r)$:

1. ...
2. ...

Fig. 12: Alternative OKVS scheme with Optimal Triangulation; Steps referenced using ... follow the corresponding ones from Figure 1.