On the Costs of Multi-Server Volume-Hiding

Amine Bahi[1], Marilyn George[2], Seny Kamara[2,3], and Tarik Moataz[2]

[1] École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France
amine.bahi@ens.psl.eu
[2] MongoDB Research, 1633 Broadway, NY 10019, USA
marilyn.george,seny.kamara,tarik.moataz@mongodb.com
[3] Brown University, Providence, RI 02912, USA

Abstract. We show that multi-server volume-hiding encrypted multi-map schemes are subject to a communication lower bound negligibly close to that of single-server schemes, even when the adversary corrupts *only a single server*. Multi-maps are data structures that map a label to a tuple of values. Volume-hiding encrypted multi-maps (Kamara and Moataz, *Eurocrypt '19*) are encrypted multi-maps that hide the length of a client's query response from an untrusted server. In the single-server setting with perfect correctness, it is known that communication for all queries must be at least t, where t is the maximum length of a tuple in the input multi-map (Kamara and Moataz, *Eurocrypt '19*, Patel et. al, *CCS '19*). In this work, we propose a *Transform-and-Split framework* for volume-hiding schemes, and show that correct multi-server schemes must have at least $t - \mathsf{negl}(k)$ communication, where k is the security parameter. On the positive side, we construct a multi-server encrypted multi-map scheme VSM that utilizes a *sharding* approach to achieve optimal communication complexity at the cost of partial volume leakage.

1 Introduction

Structured encryption (STE) is a cryptographic primitive that allows a client to outsource the storage of a data structure to an untrusted server (e.g. a cloud provider) [18]. An encrypted multi-map is a basic encrypted data structure that maps labels to tuples of values. Encrypted multi-maps are a core building block of many outsourced applications: search indices for keyword queries [38,20] and Boolean queries [13,15,26] over encrypted files, encrypted graph algorithms [33], and SQL on encrypted databases [27,42,16]. Encrypted multi-map schemes support highly efficient querying for such applications by revealing partial information about the plaintext data and queries to the untrusted server.

Leakage. The *leakage profile* of an STE scheme, consisting of individual components known as *leakage patterns*, formally describes the information revealed to the untrusted server about the client's data and queries. Starting with Islam et al.[25], there has been significant interest in using the leakage of STE schemes to extract more information about the underlying data and queries (e.g.[12], [31], [24], see [10] for a survey). The most studied and well-known leakage patterns

are the *query equality* pattern, which reveals when two client queries are equal, and the *response length* pattern, which reveals the length of the response to the client's query.

Volume-hiding. In 2018, Kamara and Moataz initiated the study of leakage suppression, or generic techniques to mitigate the common leakages that occur in STE schemes [30]. For encrypted multi-maps, the volume pattern reveals the length of the encrypted tuple returned for a client's query. As a simple volume-hiding (but storage inefficient) scheme, a client could pad all the tuples to the same length using placeholder values as padding. In 2019, Kamara and Moataz constructed the first efficient *volume-hiding* encrypted multi-map schemes, which hide the length of the client's query response [28]. Later the same year, Patel, Persiano, Yeo, and Yung formally defined volume-hiding and proposed schemes with concretely better storage and communication [36]. In 2023, Bienstock, Patel, Seo, and Yeo achieved even better concrete overheads for storage and communication in the state-of-the-art rbMM scheme [9] by utilizing a connection between volume-hiding encrypted multi-maps and Oblivious Key Value Stores (OKVS) [21]. From the first papers on volume-hiding, the lower bound on communication was known to be the length of the maximum tuple in the input multi-map. Intuitively, this was because the adversarial server would immediately observe and identify any encrypted response smaller than the length of the maximum tuple.

The costs of volume-hiding. While optimal, the communication cost of the maximum tuple length can be prohibitive in practice, when data distributions are skewed. For example, using a multi-map to index the classic Enron dataset with over 400k labels: the 99^{th} percentile tuple contains just over 2000 values, while the maximum length tuple contains over 500000 values [19]. If we wanted to hide response lengths, the lower bound implies a communication overhead of over 250x for 99% of the dataset, even for the best volume-hiding multi-map! Unfortunately, in the single-server setting we cannot reduce the costs of volume-hiding any further and we are stuck with these prohibitive overheads. On the other hand, in practical settings, we often have access to multiple servers. Then the natural next question is:

> *Can we improve the communication cost of volume-hiding in the multi-server setting?*

Surprisingly, we answer this question in the negative, even when the adversary corrupts *only one* of the servers.

1.1 Our Contributions

Definitions and framework. We extend the standard STE definitions and single-server volume-hiding definitions to the multi-server setting (Section 4). We also define a *Transform-and-Split* framework that captures all existing volume-hiding techniques, such as padding and overlapping tuples of the multi-map, and generalizes them to the multi-server setting (Section 5). Our framework explicitly embeds the transformation that volume-hiding schemes utilize to create the encrypted response for each query, thereby enabling us to bound their communication complexity.

A lower bound. We show a general lower bound on communication for volume-hiding schemes in our framework and show that *any correct* multi-server volume-hiding scheme is subject to the *same lower bound on communication* as in the single-server setting, up to a loss negligible in the security parameter. In the single-server setting, the maximum tuple length bound is easier to derive, because the server sees the total length of every encrypted response.[4] Then, it follows that any response shorter than the maximum tuple length must have less results, which breaks the volume-hiding property of the scheme. In the multi-server setting, the main challenge is that any corrupted server can see only a part of the entire encrypted response, and this partial response could contain padding (dummy results) as well as overlapping tuples (false positive results retrieved for a query). Our bound also has to account for the fact that these dummies and overlaps could be chosen randomly for each execution, for each possible input multi-map. Regardless, we can still show that if a multi-server encrypted multi-map violates our communication bound, then it cannot be volume-hiding. (Section 6).

A scheme with optimal communication. Since our lower bound rules out all existing volume-hiding techniques, we relax the strict definition of multi-server volume-hiding to allow for practical constructions with meaningful security. In particular, we construct a multi-server scheme VSM with partial volume leakage. Our scheme uses a sharding approach, common in practical distributed systems, to randomly distribute the tuples of the input multi-map across the servers. It then encrypts each random group of tuples with a basic encrypted multi-map scheme. At query time, VSM queries only the appropriate server and retrieves the queried tuple. As a result, VSM achieves optimal communication complexity, at the cost of leaking a subset of query volumes to a corrupted server (Section 7). In light of our bound, such a sharding approach might in fact be the best possible method to hide query volumes with multiple servers.

[4] Note the distinction between the length of the query response and the length of the encrypted response, which can be shorter or longer than the true response.

2 Related Work

SSE lower bounds. Early work on lower bounds for encrypted data structures was in the context of Searchable Symmetric Encryption (SSE). SSE is a cryptographic primitive that allows a client to search over a collection of documents outsourced to an untrusted server using keywords [22,38,20]. The earliest of these SSE bounds studied the trade-offs between the size of the encrypted structure, locality (number of non-contiguous reads) and read efficiency (length of the encrypted response). The first such bound was proposed by Cash and Tessaro [14], and this was followed by Asharov et al. proposing the pad-and-split framework and the statistical independence framework for existing SSE schemes [6,7]. While these frameworks are similar in spirit to our transform-and-split framework, they are focussed on deriving locality bounds, and ours is tailored to deriving communication lower bounds for volume-hiding schemes. There has also been a line of work studying the security guarantees and necessary performance trade-offs when supporting updates and more complex queries to the collection of documents (e.g. [29,39,11,26]).

STE lower bounds. Chase and Kamara [18] presented a lower bound on the token length of structured encryption schemes. Patel et al. [35] showed that the logarithmic lower bound for oblivious RAM [23] also applies to encrypted multi-map schemes that suppress the query equality leakage, and Persiano and Yeo [37] extended the result to the multi-server setting. For volume-hiding schemes, Ando and George [5] showed a bound on the storage of volume-hiding schemes in a restricted setting without a client stash.

Multi-server lower bounds. Lower bounds have also been studied in the multi-server setting for other cryptographic primitives. Notably, Larsen et al. [32] extend the ORAM communication lower bound to the multi-server setting and Yeo [41] presents lower bounds for batch Private Information Retrieval (PIR) in the multi-server setting. Persiano and Yeo [37] also present lower bounds in the multi-server setting for Differentially Private RAM (DPRAM).

Encrypted distributed data structures. Our multi-server encrypted multi-map schemes make use of *sharding*, which is a commonly used technique in distributed database systems, such as Google's Bigtable [17] and MongoDB [34]. Another recent line of work by Agarwal et al. uses techniques from distributed systems to design encrypted data structures such as dictionaries [2] and multi-maps [3,1] for use in real-world systems.

3 Preliminaries

Notation. We denote the security parameter as k, and all algorithms run in time polynomial in k. The set of all binary strings of length n is denoted as $\{0,1\}^n$, and the set of all finite binary strings as $\{0,1\}^*$. $[n]$ is the set of integers

$\{1, \ldots, n\}$, and $2^{[n]}$ is the corresponding power set. We write $x \leftarrow \chi$ to represent an element x being sampled from a distribution χ, and $x \xleftarrow{\$} X$ to represent an element x being sampled uniformly at random from a set X. The output x of an algorithm $\mathcal{A}$ is denoted by $x \leftarrow \mathcal{A}$. Given a sequence $\mathbf{v}$ of n elements, we refer to its i^{th} element as v_i or $\mathbf{v}[i]$. If S is a set then $\#S$ refers to its cardinality. If s is a string then $|s|_2$ refers to its bit length.

The word RAM. Our model of computation is the word RAM. In this model, we assume memory holds an infinite number of w-bit words and that arithmetic, logic, read and write operations can all be done in $O(1)$ time. We denote by $|x|_w$ the word-length of an item x; that is, $|x|_w = |x|_2/w$. Here, we assume that $w = \Omega(\log k)$.

Abstract data types. An *abstract data type* specifies the functionality of a data structure. It is a collection of data objects together with a set of operations defined on those objects. Examples include sets, dictionaries (also known as key-value stores or associative arrays) and graphs. The operations associated with an abstract data type fall into one of two categories: query operations, which return information about the objects; and update operations, which modify the objects. If the abstract data type supports only query operations it is *static*, otherwise it is *dynamic*.

In this paper, we consider only static data types and therefore our subsequent definitions will only consider query operations. We model a static data type $\mathbf{T}$ as a collection of three spaces: the object space $\mathbb{D} = \{\mathbb{D}_k\}_{k\in\mathbb{N}}$, the query space $\mathbb{Q} = \{\mathbb{Q}_k\}_{k\in\mathbb{N}}$, and the response space $\mathbb{R} = \{\mathbb{R}_k\}_{k\in\mathbb{N}}$. We also define the query map $\mathsf{qu} : \mathbb{D} \times \mathbb{Q} \to \mathbb{R}$ to represent query operations associated with the data type. When specifying a data type $\mathbf{T}$ we will often just describe its query map qu from which the object, query, and response spaces can be deduced. The spaces are ensembles of finite sets of finite strings indexed by the security parameter. We assume that $\mathbb{R}$ includes a special element $\perp$ and that $\mathbb{D}$ includes an empty object d_0 such that for all $q \in \mathbb{Q}$, $\mathsf{qu}(d_0, q) = \perp$.

Data structures. A type-$\mathbf{T}$ data *structure* is a representation of data objects in $\mathbb{D}$ in some computational model (here we use the word RAM). Typically, the representation is optimized to support qu as efficiently as possible; that is, such that there exists an efficient algorithm Query that computes the function qu.

Definition 1 (Structuring scheme). *Let* $\mathbf{T} = (\mathsf{qu} : \mathbb{D} \times \mathbb{Q} \to \mathbb{R})$ *be a static type. A type-*$\mathbf{T}$ *structuring scheme* $\mathsf{SS} = (\mathsf{Setup}, \mathsf{Query})$ *is composed of two polynomial-time algorithms that work as follows:*

- $\mathsf{DS} \leftarrow \mathsf{Setup}(d)$*: is a possibly probabilistic algorithm that takes as input a data object* $d \in \mathbb{D}$ *and outputs a data structure* DS*. Note that* d *can be represented in any arbitrary manner as long as its bit length is polynomial in* k*. Unlike* DS*, its representation does not need to be optimized for any particular query.*
- $r \leftarrow \mathsf{Query}(\mathsf{DS}, q)$*: is an algorithm that takes as input a data structure* DS *and a query* $q \in \mathbb{Q}$ *and outputs a response* $r \in \mathbb{R}$*.*

Here, we allow Setup to be probabilistic but not Query. This captures most data structures but the definition can be extended to include structuring schemes with probabilistic query algorithms. We say that a data structure DS *instantiates* a data object $d \in \mathbb{D}$ if for all $q \in \mathbb{Q}$, $\mathsf{Query}(\mathsf{DS}, q) = \mathsf{qu}(d, q)$. We denote this by $\mathsf{DS} \equiv d$. We denote the set of queries supported by a structure DS as $\mathbb{Q}_{\mathsf{DS}}$; that is,

$$\mathbb{Q}_{\mathsf{DS}} \stackrel{def}{=} \Big\{ q \in \mathbb{Q} : \mathsf{Query}(\mathsf{DS}, q) \neq \perp \Big\}.$$

Similarly, the set of responses supported by a structure DS is denoted $\mathbb{R}_{\mathsf{DS}}$.

Definition 2 (Correctness). *Let* $\mathbf{T} = (\mathsf{qu} : \mathbb{D} \times \mathbb{Q} \to \mathbb{R})$ *be a static type. A type-*$\mathbf{T}$ *structuring scheme* $\mathsf{SS} = (\mathsf{Setup}, \mathsf{Query})$ *is perfectly correct if for all* $d \in \mathbb{D}$*,*

$$\Pr\left[\, \mathsf{DS} \equiv d : \mathsf{DS} \leftarrow \mathsf{Setup}(d) \,\right] = 1,$$

where the probability is over the coins of Setup*.*

Basic multi-maps. In this paper we consider a basic data type known as a multi-map, which we recall here. A multi-map structure MM is a collection of label/tuple pairs $\{(\ell_i, \mathbf{v}_i)_i\}_{i \leq n}$ that supports get and put operations. We write $\mathbf{v}_i := \mathsf{MM}[\ell_i]$ to denote getting the tuple associated with label ℓ_i and $\mathsf{MM}[\ell_i] := \mathbf{v}_i$ to denote operation of associating the tuple $\mathbf{v}_i$ to label ℓ_i. We use $\#\mathsf{MM}[\ell_i]$ to denote the number of values in the tuple $\mathsf{MM}[\ell_i]$ and $\#\mathsf{MM}$ to denote the total number of values in MM. Multi-maps are the abstract data type instantiated by an inverted index. In the encrypted search literature multi-maps are sometimes referred to as indexes, databases or tuple-sets (T-sets).

Leakage. We use the following leakage patterns in this work. Let $\mathbf{T} = (\mathsf{qu} : \mathbb{D} \times \mathbb{Q} \to \mathbb{R})$ be a static type.

- the *data size pattern* is the function family $\mathsf{dsize} = \{\mathsf{dsize}_k\}_{k \in \mathbb{N}}$ with $\mathsf{dsize}_k : \mathbb{D}_k \to \mathbb{N}$ such that $\mathsf{dsize}_k(d) = |d|_w$;
- the *query equality pattern* is the function family $\mathsf{qeq} = \{\mathsf{qeq}_{k,t}\}_{k,t \in \mathbb{N}}$ with $\mathsf{qeq}_{k,t} : \mathbb{D}_k \times \mathbb{Q}_k^t \to \{0,1\}^{t \times t}$ such that $\mathsf{qeq}_{k,t}(d, q_1, \ldots, q_t) = M$, where M is a binary $t \times t$ matrix such that for queries q_i and q_j, $M[i,j] = 1$ if $q_i = q_j$ and $M[i,j] = 0$ otherwise;
- The *response length pattern* is the function family $\mathsf{rlen} = \{\mathsf{rlen}_{k,t}\}_{k,t \in \mathbb{N}}$ with $\mathsf{rlen}_{k,t} : \mathbb{D}_k \times \mathbb{Q}_k^t \to \mathbb{N}$ such that $\mathsf{rlen}_{k,t}(d, q_1, \ldots, q_t) = (|\mathsf{qu}(d, q_1)|, \ldots, |\mathsf{qu}(d, q_t)|)$.

Cryptographic protocols. We denote by $(\mathsf{out}_A, \mathsf{out}_B) \leftarrow \Pi_{A,B}(X, Y)$ the execution of a two-party protocol Π between parties A and B, where X and Y are the inputs provided by A and B, respectively; and out_A and out_B are the outputs returned to A and B, respectively.

3.1 Structured Encryption

Definition 3 (Structured encryption [18]). *An interactive static structured encryption scheme* $\Sigma = (\mathsf{Setup}, \mathsf{Query}_{\mathbf{C},\mathbf{S}})$ *consists of an algorithm and a two-party protocol that work as follows:*

- $(K, st, \mathsf{EDS}) \leftarrow \mathsf{Setup}(1^k, \mathsf{DS})$*: is a probabilistic polynomial-time algorithm that takes as input a security parameter* 1^k *and a type-*$\mathbf{T}$ *structure* DS*. It outputs a secret key* K*, a state* st *and an encrypted structure* EDS*. If* $\mathsf{DS} \equiv d_0$*, it outputs an empty* EDS*.*
- $\big((st', r), \mathsf{EDS}'\big) \leftarrow \mathsf{Query}_{\mathbf{C},\mathbf{S}}\big((K, st, q), \mathsf{EDS}\big)$*: is a two-party protocol executed between a client and a server where the client inputs a secret key* K*, a state* st *and a query* q *and the server inputs an encrypted structure* EDS*. The client receives as output a (possibly) updated state* st' *and a response* $r \in \mathbb{R} \cup \perp$ *while the server receives a (possibly updated) encrypted structure* EDS'*.*

Definition 4 (Security [20,18]). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Query}_{\mathbf{C},\mathbf{S}})$ *be a structured encryption scheme and consider the following probabilistic experiments where* $\mathcal{C}$ *is a stateful challenger,* $\mathcal{A}$ *is a stateful adversary,* $\mathcal{S}$ *is a stateful simulator,* $\Lambda = (\mathsf{patt}_{\mathsf{S}}, \mathsf{patt}_{\mathsf{Q}})$ *is a leakage profile and* $z \in \{0,1\}^*$*:*

$\mathbf{Real}_{\Sigma,\mathcal{C},\mathcal{A}}(k)$*: given* z *the adversary* $\mathcal{A}$ *outputs a structure* DS *and receives* EDS *from the challenger, where* $(K, st, \mathsf{EDS}) \leftarrow \mathsf{Setup}(1^k, \mathsf{DS})$*.* $\mathcal{A}$ *then adaptively chooses a polynomial-size sequence of queries* $(q_1, \ldots q_m)$*. For all* $1 \leq i \leq m$ *the challenger and adversary do the following:*
- *execute* $\mathsf{Query}_{\mathcal{C},\mathcal{A}}\big((K, st, q_i), \mathsf{EDS}\big)$*.*

Finally, $\mathcal{A}$ *outputs a bit* b *that is output by the experiment.*

$\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k)$*: given* z *the adversary* $\mathcal{A}$ *outputs a structure* DS *of type* $\mathbf{T}$*. Given* $\mathsf{patt}_{\mathsf{S}}(\mathsf{DS})$*, the simulator returns an encrypted structure* EDS *to* $\mathcal{A}$*.* $\mathcal{A}$ *then adaptively chooses a polynomial-size sequence of queries* $(q_1, \ldots, q_m)$*. For all* $1 \leq i \leq m$*, the simulator and adversary do the following:*
- $\mathcal{S}$ *is given* $\mathsf{patt}_{\mathsf{Q}}(\mathsf{DS}, \mathbf{q}_1, \ldots, \mathbf{q}_i)$ *and it executes* $\mathsf{Query}_{\mathcal{A},\mathcal{S}}$ *with* $\mathcal{A}$

Finally, $\mathcal{A}$ *outputs a bit* b *that is output by the experiment.*

We say that Σ *is* Λ*-secure if there exists a PPT simulator* $\mathcal{S}$ *such that for all PPT adversaries* $\mathcal{A}$*, for all* $z \in \{0,1\}^*$*,*

$$|\Pr[\mathbf{Real}_{\Sigma,\mathcal{C},\mathcal{A}}(k) = 1] - \Pr[\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k) = 1]| \leq \mathsf{negl}(k).$$

4 Definitions

We now extend the standard STE syntax and security definitions to the multi-server setting with n servers, $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n$. The multi-server syntax allows for n encrypted data structures and a multi-party Query protocol.

Definition 5 (Multi-server structured encryption). *An interactive static multi-server structured encryption scheme* $\Sigma = (\mathsf{Setup}, \mathsf{Query}_{\mathbf{C},\mathbf{S}_1,\ldots,\mathbf{S}_n})$ *consists of an algorithm and a multi-party protocol that work as follows:*

– $(K, st, (\mathsf{EDS}_1, \ldots, \mathsf{EDS}_n)) \leftarrow \mathsf{Setup}(1^k, n, \mathsf{DS})$*: is a probabilistic polynomial-time algorithm that takes as input a security parameter* 1^k*, the number of servers* n*, and a type-*$\mathbf{T}$ *structure* DS*. It outputs a secret key* K*, a state* st *and* n *encrypted structures* $\mathsf{EDS}_1, \ldots, \mathsf{EDS}_n$*. If* $\mathsf{DS} \equiv d_0$*, it outputs* n *empty encrypted structures.*
– $\big((st', r), \mathsf{EDS}'_1, \ldots, \mathsf{EDS}'_n\big) \leftarrow \mathsf{Query}_{\mathbf{C},\mathbf{S}_1,\ldots,\mathbf{S}_n}\big((K, st, q), \mathsf{EDS}_1, \ldots, \mathsf{EDS}_n\big)$*: is a multi-party protocol executed between a client and* n *servers where the client inputs a secret key* K*, a state* st *and a query* q *and server* $\mathbf{S}_i$ *inputs an encrypted structure* EDS_i*. The client receives as output a (possibly) updated state* st' *and a response* $r \in \mathbb{R} \cup \perp$ *while server* $\mathbf{S}_i$ *receives a (possibly updated) encrypted structure* EDS'_i*.*

Security. In the multi-server security definition, we allow the adversary to corrupt any subset of the servers $\mathrm{C} \subseteq [n]$ up to a corruption threshold $\#\mathrm{C} \leq c$. We also define up to n distinct leakage profiles $\Lambda^1, \ldots, \Lambda^n$, one for each possible corrupted server, and the simulator receives the leakages corresponding to the corrupted servers C in the ideal world.

Definition 6 (c-server Λ-security). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{Query}_{\mathbf{C},\mathbf{S}_1,\ldots,\mathbf{S}_n})$ *be a multi-server structured encryption scheme and consider the following probabilistic experiments where* $\mathcal{C}$ *is a stateful challenger,* $\mathcal{A}$ *is a stateful adversary,* $\mathcal{S}$ *is a stateful simulator, and* $\Lambda = \{\Lambda^i,\ i \in [n]\}$ *where* $\Lambda^i = (\mathsf{patt}^i_{\mathsf{S}}, \mathsf{patt}^i_{\mathsf{Q}})$ *for* $1 \leq i \leq n$ *are* n *leakage profiles,* $c \leq n$ *is a corruption threshold, and* $z \in \{0,1\}^*$*:*

$\mathbf{Real}_{\Sigma,\mathcal{C},\mathcal{A}}(k)$*: given* z*, the adversary* $\mathcal{A}$ *outputs a subset* $\mathrm{C} \subseteq [n]$ *and a structure* DS *and receives* EDS_j *for all* j *in* C *from the challenger* $\mathcal{C}$*, where*

$$(K, st, (\mathsf{EDS}_1, \ldots, \mathsf{EDS}_n)) \leftarrow \mathsf{Setup}(1^k, n, \mathsf{DS}).$$

$\mathcal{A}$ then adaptively chooses a polynomial-size sequence of queries $(q_1, \ldots q_m)$. For all $1 \leq i \leq m$ the challenger and adversary do the following:
– *execute* $\mathsf{Query}_{\mathcal{C},\mathcal{A}}\big((K, st, q_i), (\mathsf{EDS}_j, j \in \mathrm{C})\big)$*. $\mathcal{A}$ receives the views of the corrupted servers $\mathbf{S}_i$ for $i \in \mathrm{C}$ and interacts with $\mathcal{C}$ who represents the honest parties.*

Finally, $\mathcal{A}$ outputs a bit b that is output by the experiment.

$\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k)$*: given* z*, the adversary* $\mathcal{A}$ *outputs a subset* $\mathrm{C} \subseteq [n]$ *and a structure* DS*. Given* $\mathsf{patt}^j_{\mathsf{S}}(\mathsf{DS})$*, the simulator returns the encrypted structures* EDS^j*, for all* $j \in \mathrm{C}$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *then adaptively chooses a polynomial-size sequence of queries* $(q_1, \ldots, q_m)$*. For all* $1 \leq i \leq m$*, the simulator and adversary do the following:*
– *$\mathcal{S}$ is given $\mathsf{patt}^j_{\mathsf{Q}}(\mathsf{DS}, q_1, \ldots, q_i)$, for all $j \in \mathrm{C}$ and it executes $\mathsf{Query}_{\mathcal{A},\mathcal{S}}$ with $\mathcal{A}$, where $\mathcal{A}$ interacts with $\mathcal{S}$ as the corrupted servers $\mathbf{S}_i$ for $i \in \mathrm{C}$.*

Finally, $\mathcal{A}$ outputs a bit b that is output by the experiment.

We say that Σ *is* c*-server* Λ*-secure if there exists a* PPT *simulator* $\mathcal{S}$ *such that for all* PPT *adversaries* $\mathcal{A}$*, for all* $\mathrm{C} \subseteq [n]$ *such that* $\#\mathrm{C} \leq c$*, for all* $z \in \{0,1\}^*$*,*

$$|\Pr[\mathbf{Real}_{\Sigma,\mathcal{C},\mathcal{A}}(k) = 1] - \Pr[\mathbf{Ideal}_{\Sigma,\mathcal{A},\mathcal{S}}(k) = 1]| \leq \mathsf{negl}(k).$$

4.1 Multi-Server Volume-Hiding

We now define volume-hiding leakage profiles for encrypted multi-map schemes in the multi-server setting. Our definition generalizes the single-server volume-hiding definition presented by Patel, Persiano, Yeo, and Yung in 2019. Note that both the single-server and multi-server definitions specify properties of *leakage profiles*, not the STE schemes themselves. An STE scheme is volume-hiding if it has such a volume-hiding leakage profile. We recall the definition of the signature of a multi-map:

Definition 7 (Signature of a multi-map [36]). *The signature of a multi-map* MM *is the sequence of pairs* $((\ell, \#\mathsf{MM}(\ell)))_{\ell \in \mathbb{L}}$ *where* $\#\mathsf{MM}(\ell)$ *is the length of the tuple of values* $\mathsf{MM}(\ell)$ *associated with the label* ℓ *in the multi-map.*

At a high level, we generalize the single-server definition by allowing for n servers $\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n$. The adversary in the multi-server game corrupts some subset of servers $\mathrm{C} \in [n]$. Then a multi-server leakage profile is volume-hiding if the adversary cannot distinguish the leakage of two different multi-map signatures subject to constraints on the total size and the maximum tuple length of the two multi-maps.

Formally, we define the game $\mathbf{MSVHGame}^{\mathcal{A},\Lambda}_{\eta,\mathrm{C}}(N,n,t)$ for a multi-server leakage profile Λ where $\Lambda^i = (\mathsf{patt}^i_{\mathsf{S}}, \mathsf{patt}^i_{\mathsf{Q}})$, an adversary $\mathcal{A}$, a corrupted subset $\mathrm{C} \subseteq [n]$, and $\eta \in \{0,1\}$.

$\mathbf{MSVHGame}^{\mathcal{A},\Lambda}_{\eta,\mathrm{C}}(N,n,t)$:

1. $\mathcal{A}$ generates two signatures $\mathsf{sign}_0 = ((\ell, \#\mathsf{MM}_0(\ell)))_{\ell \in \mathbb{L}}$ and $\mathsf{sign}_1 = ((\ell, \#\mathsf{MM}_1(\ell)))_{\ell \in \mathbb{L}}$ such that:
   - $\sum_{\ell \in \mathbb{L}} \#\mathsf{MM}_0(\ell) = \sum_{\ell \in \mathbb{L}} \#\mathsf{MM}_1(\ell) = N$;
   - $\max_{\ell \in \mathbb{L}} \#\mathsf{MM}_0(\ell) = \max_{\ell \in \mathbb{L}} \#\mathsf{MM}_1(\ell) = t$.
2. The challenger $\mathcal{C}$ receives signatures sign_0 and sign_1 from the adversary $\mathcal{A}$ and generates a multi-map MM with the signature sign_η. Specifically, the challenger generates $\#\mathsf{MM}_\eta(\ell)$ arbitrary values for each label $\ell \in \mathbb{L}$. The challenger then sends $\mathsf{patt}^i_{\mathsf{S}}(\mathsf{MM})$ to the adversary $\mathcal{A}$ for all i in C.
3. The adversary adaptively picks labels $\ell_1, \ell_2, \cdots$ for query operations. For each label ℓ_j, for all $i \in \mathrm{C}$, the challenger $\mathcal{C}$ computes $\mathsf{patt}^i_{\mathsf{Q}}(\mathsf{MM}, \ell_1, \cdots, \ell_j)$ and sends it to the adversary.
4. Finally, the adversary outputs a bit $b \in \{0,1\}$.

We denote by $p^{\mathcal{A},\Lambda}_{\eta,\mathrm{C}}(N,n,t)$ the probability that $\mathcal{A}$ outputs 1 when playing game $\mathbf{MSVHGame}^{\mathcal{A},\Lambda}_{\eta,\mathrm{C}}(N,n,t)$.

Definition 8 (c-server volume-hiding leakage). *A multi-server leakage profile* Λ *is* c*-server volume-hiding if and only if for all adversaries* $\mathcal{A}$*, for* $n \geq 1$*, and for all* $N \geq t \geq 1$*, for all* $\mathrm{C} \subseteq [n]$ *such that* $\#\mathrm{C} \leq c$*,*

$$p^{\mathcal{A},\Lambda}_{0,\mathrm{C}}(N,n,t) = p^{\mathcal{A},\Lambda}_{1,\mathrm{C}}(N,n,t).$$

We can recover the definition of a single-server volume-hiding leakage profile by setting $n = 1$. If a multi-server multi-map encryption scheme Σ is c-server Λ-secure and Λ is a c-server volume-hiding leakage profile, then we say that the scheme Σ is a c-server volume-hiding multi-map encryption scheme.

5 The Transform-and-Split Framework

We now describe our transform-and-split framework for multi-server encrypted multi-map schemes. As the name suggests, this framework captures schemes that first transform the input data structure and then split the resulting structure over multiple servers. Existing single-server encrypted multi-map schemes already incorporate data transforms explicitly or implicitly. In volume-hiding schemes in particular, the transformation step includes padding or overlapping the tuples of the input multi-map. Our framework adds an explicit split step to model multi-server schemes that distribute the encrypted data structures over multiple servers. More formally:

- $(\mathsf{TMM}, \mathsf{tmap}) \leftarrow \mathsf{Transform}(\mathsf{MM}, \mathsf{params}, n)$: a (possibly randomized) algorithm that takes as input a multi-map MM, global parameters params, and the number of servers n and outputs a transformed multi-map TMM, and a transform map tmap that stores the state of the transform.
- $(\mathsf{TMM}_1, \ldots \mathsf{TMM}_n, \mathsf{smap}) \leftarrow \mathsf{Split}(\mathsf{TMM}, \mathsf{tmap}, n)$: a (possibly randomized) algorithm that takes as input a transformed multi-map TMM, a transform map tmap and the number of servers n and output split multi-maps TMM_1, $\ldots, \mathsf{TMM}_n$ and a split map smap that stores the state of the split.

When it is helpful to make randomness explicit, we write these algorithms as $\mathsf{Transform}(\mathsf{MM}, \mathsf{params}, n; \mathbf{r}_\mathsf{T})$ and $\mathsf{Split}(\mathsf{TMM}, \mathsf{tmap}, n; \mathbf{r}_\mathsf{S})$ where $\mathbf{r}_\mathsf{T}$ and $\mathbf{r}_\mathsf{S}$ denote their respective random coins.

Overview. At a high-level, $\mathsf{Transform}$ takes each label-tuple pair in the input MM and transforms it into one or more label-tuple pairs, adding padding or overlapping tuples as required by the scheme. The Split algorithm then distributes these transformed tuples across the n servers. The corresponding state maps tmap and smap store the information required to reconstruct the transformation and the splitting at query time. During a query, the client uses tmap and smap to identify the transformed labels and the servers that must be queried to retrieve the values for the queried label.

Transforming the input multi-map. As a simple example of a transformed multi-map, we will use the first (single-server) volume-hiding scheme VLH, proposed by Kamara and Moataz [28] to illustrate the components of our framework. The VLH scheme uses an underlying data transform known as the pseudo-random transform (PRT). The PRT uses two parameters: λ and ν, which determine the length of the transformed tuples. For each label ℓ, PRT samples a pseudo-random length len from the interval $[0, \nu]$ for every tuple, and sets the tuple length to be $n'_\ell := \lambda + \mathsf{len}$. The input tuple is then either padded or truncated to length n'_ℓ, and the resulting multi-map is the transformed multi-map. In our framework, this output multi-map MM' of the PRT corresponds to TMM, and tmap maps each original label to its corresponding labels in TMM. In this particular example, tmap simply maps every label to itself.

Splitting the transformed multi-map. While existing volume-hiding schemes do not use multiple servers, a multi-server scheme would require an additional algorithm to split the transformed multi-map across the servers. As a simple example, consider a two-server variant of VLH with servers $\mathbf{S}_1$ and $\mathbf{S}_2$. Let Split construct TMM_1 from the tuples in the first half of TMM and TMM_2 from the remaining tuples. Then smap would map the labels in the first half to $\mathbf{S}_1$ and the remaining labels to $\mathbf{S}_2$. Looking ahead, the client $\mathbf{C}$ stores tmap and smap as part of its local state and uses them during the Query protocol to issue queries on the appropriate transformed labels to the servers that store them.

Constraints. We require that any truncation of the tuples, or removal of values in the input MM, occur during Transform. It is straightforward to see that if a scheme needs to remove values during or after the splitting step, this can be simulated inside Transform and reflected in the output TMM. In contrast, both Transform and Split are allowed to add values to the input multi-map. In particular, this allows schemes in our framework to add server-dependent padding to each split multi-map during Split. For correctness, we assume that the client has access to an explicit StripResponse algorithm that removes the irrelevant padding and results from the response. We now describe the structure of a generic encrypted multi-map scheme within our framework.

5.1 A Generic Transform-and-Split Scheme

The pseudocode for a transform-and-split encrypted multi-map scheme is given in Figure 1. The scheme uses a multi-map encryption scheme Σ_{MM} as a black-box. For example, Σ_{MM} could be the Π_{bas} scheme of Cash et al. with no dynamism [15]. The response-hiding version of this basic scheme leaks the total size of the input multi-map at setup, and the query equality and volume at query time.

Description. Our framework uses a multi-map encryption scheme Σ_{MM} together with the Transform and Split algorithms to construct a multi-server encrypted multi-map scheme Σ_{TS} as follows:

- Setup. Given an input multi-map MM, Setup first runs Transform to obtain a transformed multi-map TMM and a helper data structure tmap. It then runs Split on $(\mathsf{TMM}, \mathsf{tmap})$ with respect to the number of servers n, yielding transformed-and-split multi-maps TMM_i for each server $\mathbf{S}_i$. Next, Setup encrypts each TMM_i using Σ_{MM} to obtain an encrypted multi-map for each server. Finally, it outputs the keys for each encrypted multi-map, the client state (including the helper structures tmap and smap), and the n encrypted multi-maps for the n servers.
- Query. In this protocol, the client uses the helper structures tmap and smap to query the transformed-and-split encrypted multi-maps and recover the response. It first parses the key and client state. Then, given a query q, the client retrieves from tmap the tuple τ of transformed labels associated with q. For each transformed label $q_{\mathsf{T}} \in \tau$, it uses $\mathsf{smap}[q_{\mathsf{T}}]$ to obtain the list

Given a multi-map encryption scheme $\Sigma_{\mathsf{MM}} = (\mathsf{Setup}, \mathsf{Get})$ and $\mathsf{Transform}$ and Split algorithms with parameters params as described in our framework, we define a generic transform-and-split scheme $\Sigma_{\mathsf{TS}} = (\mathsf{Setup}, \mathsf{Query})$ as follows:

- $(K, st, \mathsf{EMM}_1, \ldots, \mathsf{EMM}_n) \leftarrow \mathsf{Setup}(1^k, n, \mathsf{params}, \mathsf{MM})$:
  1. run $(\mathsf{TMM}, \mathsf{tmap}) \leftarrow \mathsf{Transform}(\mathsf{MM}, \mathsf{params}, n)$;
  2. run $(\mathsf{TMM}_1, \ldots \mathsf{TMM}_n, \mathsf{smap}) \leftarrow \mathsf{Split}(\mathsf{TMM}, \mathsf{tmap}, n)$;
  3. for $i = 1 \ldots n$:
     (a) run $(K_i, st_i, \mathsf{EMM}_i) \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Setup}(1^k, \mathsf{TMM}_i)$;
  4. set $K := \{K_i, 1 \leq i, \leq n\}$;
  5. set $st := \big(\{st_i, 1 \leq i \leq n\}, \mathsf{tmap}, \mathsf{smap}\big)$;
  6. output $(K, st, \mathsf{EMM}_1, \ldots, \mathsf{EMM}_n)$.
- $((st', r), \mathsf{EMM}'_1, \ldots, \mathsf{EMM}'_n) \leftarrow \mathsf{Query}_{\mathbf{C}, \mathbf{S}_1, \ldots, \mathbf{S}_n}((K, st, q), \mathsf{EMM}_1, \ldots, \mathsf{EMM}_n)$:
  1. $\mathbf{C}$ initializes r to be empty;
  2. $\mathbf{C}$ parses K as $\{K_i, 1 \leq i, \leq n\}$;
  3. $\mathbf{C}$ parses st as $\big(\{st_i, 1 \leq i \leq n\}, \mathsf{tmap}, \mathsf{smap}\big)$;
  4. $\mathbf{C}$ sets $\mathbf{t} := \mathsf{tmap}[q]$;
  5. for each q_{T} in $\mathbf{t}$:
     - for each $\mathbf{S}_j$ in $\mathsf{smap}[q_{\mathsf{T}}]$:
       (a) $\mathbf{C}$ and $\mathbf{S}_j$ run
       $$((st'_j, r_j), \mathsf{EMM}'_j) \leftarrow \Sigma_{\mathsf{MM}}.\mathsf{Query}_{\mathbf{C}, \mathbf{S}_j}((K_j, st_j, q_{\mathsf{T}}), \mathsf{EMM}_j);$$
       (b) $\mathbf{C}$ sets $st_j := st'_j$;
       (c) $\mathbf{S}_j$ sets $\mathsf{EMM}_j := \mathsf{EMM}'_j$;
       (d) $\mathbf{C}$ sets $r := r \cup r_j$;
  6. $\mathbf{C}$ sets $st := \big(\{st_i, 1 \leq i \leq n\}, \mathsf{tmap}, \mathsf{smap}\big)$;
  7. $\mathbf{C}$ sets $r := \mathsf{StripResponse}(r, q)$;
  8. $\mathbf{C}$ outputs (st, r) and each server $\mathbf{S}_i$ outputs EMM_i.

Fig. 1. A generic scheme in the transform-and-split framework

of servers, and queries each such server using the underlying scheme Σ_{MM}. Finally, the client combines the responses from all servers, and removes the padding and false positive values to produce the final response r. The client state remains unchanged, and each server $\mathbf{S}_i$ outputs EMM_i.

5.2 Volume-Hiding in Our Framework

In this section, we show how our framework captures existing static single-server volume-hiding schemes, as well as simple multi-server volume-hiding schemes. For all the single-server schemes, it suffices to explain how $\mathsf{Transform}$ works and assume a trivial Split algorithm that sets $\mathsf{TMM}_1 = \mathsf{TMM}$.

The VLH *scheme of Kamara and Moataz [28].* As we already discussed, our framework captures VLH directly, by setting $\mathsf{Transform}$ to be the $\mathsf{PRT.Setup}$ algorithm, params to be λ, and Σ_{MM} to be Σ_{EMM}. The total storage is then $\#\mathsf{TMM}$, and the total communication for a label ℓ is $\#\mathsf{TMM}[\ell]$.

The AVLH *scheme of Kamara and Moataz [28].* Similar to VLH, AVLH uses an underlying data transform known as the dense subgraph transform (or DST). At a high level, the DST considers each value a ball thrown into a bin. Each label ℓ is mapped to exactly t bins chosen by a pseudorandom function, and the values from the tuple $\mathsf{MM}[\ell]$ are stored in those t bins.[5] Finally, each bin is padded up to the size of the maximum bin. The transform outputs a dictionary DX that stores the bins, and a multi-map MM_G that stores the randomness required to recreate the label to bin mappings. This maps directly to our Transform algorithm, where TMM is DX and MM_G is the client state tmap used to reconstruct the transformed labels. Each label is *transformed* into t bin labels and the AVLH scheme is also captured in our framework. The total storage is then dsize(TMM), the space required to store the bins, and the total communication for a label ℓ is $\#\mathsf{TMM}[\ell_{\mathsf{T}}]$ for the t transformed labels ℓ_{T}.

The vhMM *scheme of Patel et al [36].* While the original description of vhMM does not have an explicit Transform algorithm, we can write a Transform that encapsulates it in our framework. At a high level, the scheme constructs two cuckoo hash tables T_1, T_2 of size $(1+\alpha)\cdot\#\mathsf{MM}$ and performs the cuckoo hashing insertion with the values of each tuple. Any empty slots in the tables are filled with padding. We can view this as a Transform algorithm that performs the cuckoo hashing allocation and padding and a TMM that contains both T_1 and T_2. At query time, vhMM retrieves t locations (ℓ in the original scheme) from each table to find the correct encrypted response. We can view this as a transformation of each queried label to $2 \cdot t$ labels to be queried in TMM. The total storage is then dsize(TMM), the space required to store the hash tables, and the total communication for a label ℓ is $\#\mathsf{TMM}[\ell_{\mathsf{T}}]$ for the $2 \cdot t$ transformed labels ℓ_{T}.

The dprfMM *scheme of Patel et al [36].* Similar to vhMM, dprfMM also uses two cuckoo hash tables, with the difference that it uses a delegatable PRF to compute the transformed labels to be queried. For this reason it has better communication complexity than vhMM while the total encrypted response length is still $2t$. Since the transformation step is unchanged, this scheme can also be captured in our framework but with $O(1)$ communication for querying the $2t$ transformed labels.

The xorMM *scheme of Wang et al [40].* xorMM uses an XOR filter to encode the label-value pairs in the input MM. Any empty slots in the XOR filter are padded with random values at setup. Viewing the scheme in our framework, we can set TMM to be the output EMM after the mapping of the XOR filter, and the transformed labels to be the queries made to the filter during the query protocol. The total storage is then dsize(TMM), the space required to store the XOR filter, and the total communication for a label ℓ is $\#\mathsf{TMM}[\ell_{\mathsf{T}}]$ for the $3t$ transformed labels ℓ_{T}.

[5] If the multi-map contains what is called a *concentrated component*, the labels that are part of this component are treated differently, but at the end of the transform these labels are also mapped to t bins.

The rbMM *scheme of Bienstock et al [9].* rbMM is the current state-of-the-art in static single-server volume-hiding. The scheme uses an oblivious key value store (OKVS, [21]) to encode the input label-value pairs. Any 'empty' slots in the encoding are implicitly filled with random values by the OKVS, allowing the encrypted structure to be compact. Viewing the scheme in our framework, we can set TMM to be the output EMM of the OKVS encoding, and the transformed labels to be the queries made to the OKVS during decoding. The total storage is then dsize(TMM), the space required to store the OKVS, and the total communication for a label ℓ is $\#\mathsf{TMM}[\ell_{\mathsf{T}}]$ for the t transformed labels ℓ_{T}.

Naive multi-server volume-hiding. While there are no existing multi-server volume-hiding schemes, we consider naive schemes that split the storage and communication of a single-server scheme over multiple servers. For example, we could store each hash table for vhMM or dprfMM on a different server. While these constructions can be captured in our framework by having the Split algorithm separate the multi-map, they are not technically very interesting since they do not gain total storage or communication over the single-server schemes.

6 A Lower Bound for Multi-Server Volume-Hiding

In this section we present our lower bound for multi-server volume-hiding schemes. In particular, we show that a correct multi-server volume-hiding scheme Σ must have an expected encrypted response length $\Omega(t)$ across the n servers in order to answer *any query*, where t is the maximum tuple length of an input multi-map (Corollary 1). We first prove a more general theorem that shows that any such scheme Σ must have an expected encrypted response length $\Omega(\tau^*)$ where τ^* is the smallest expected response length for an input tuple of maximum tuple length (Theorem 1).

In the following theorem, let $\mathbb{D}_{N,t}$ denote the set of all input multi-maps which have N total values and a maximum tuple length of t.

Theorem 1 (Communication Lower Bound). *Any multi-server encrypted multi-map scheme Σ_{MM} in the transform-and-split model with a volume-hiding leakage profile $\mathcal{L}_{\Sigma}$, must have expected communication complexity $\Omega(\tau^*)$, for any $n \geq 1$, for all $N \geq t \geq 1$, for any* C *such that* $\#\mathrm{C} \geq 1$*, where*

$$\tau^* \stackrel{def}{=} \min_{\mathsf{MM} \in \mathbb{D}_{N,t}} E\left[X^{t,\mathsf{sign}_{\mathsf{MM}}} \right];$$

where $X^{t,\mathsf{sign}_{\mathsf{MM}}}$ is the random variable denoting the total length of the encrypted response for an input tuple of length t when the signature of the input multi-map is $\mathsf{sign}_{\mathsf{MM}}$,[6] *and the expectation is computed over the randomness of the* Transform *and* Split *algorithms.*

[6] Note that here we assume that the length of the encrypted response only depends on the input tuple lengths, according to the definition of the security game.

Proof. We prove the theorem by setting up the following contradiction: if there exists some input multi-map $\mathsf{MM}_0 \in \mathbb{D}_{N,t}$ and tuple length len_0 such that $\mathsf{len}_0 < t$ and the (expected) total length of the encrypted response is non-negligibly smaller than τ^*, then $\mathcal{L}_\Sigma$ cannot be a volume-hiding leakage function.[7]

Let sign_0 be the signature of such a multi-map MM_0, and let ℓ_0 be the label corresponding to the tuple with length len_0. We now define the random variables $X_i^{\mathsf{len}_0,\mathsf{sign}_0}$ for $1 \leq i \leq n$, denoting the length of the encrypted (sub-)response on server $\mathbf{S}_i$ for label ℓ_0. Formally, for any $0 \leq j \leq N$,[8]

$$\Pr\left[X_i^{\mathsf{len}_0,\mathsf{sign}_0} = j \right] = \Pr\left[\left(\sum_{\ell_T \in \mathsf{tmap}[\ell_0]} \#\mathsf{TMM}_i[\ell_\mathsf{T}] \right) = j \right],$$

where $(\mathsf{TMM}, \mathsf{tmap}) \leftarrow \mathsf{Transform}(\mathsf{MM}_0, n; \mathbf{r}_\mathsf{T})$ and $(\mathsf{TMM}_1, \ldots \mathsf{TMM}_n, \mathsf{smap}) \leftarrow \mathsf{Split}(\mathsf{TMM}, \mathsf{tmap}, n; \mathbf{r}_\mathsf{S})$. We can now define the random variable $X^{\mathsf{len}_0,\mathsf{sign}_0}$ to denote the total length of the encrypted response for label ℓ_0 in sign_0 as:

$$X^{\mathsf{len}_0,\mathsf{sign}_0} = \sum_{i=1}^{n} X_i^{\mathsf{len}_0,\mathsf{sign}_0}, \text{ and therefore,}$$

$$E\left[X^{\mathsf{len}_0,\mathsf{sign}_0} \right] = \sum_{i=1}^{n} E\left[X_i^{\mathsf{len}_0,\mathsf{sign}_0} \right]. \tag{1}$$

Recall that the expected total response length for len_0 in sign_0 is non-negligibly smaller than τ^*. More formally, if:

$$\tau_0 \stackrel{def}{=} E\left[X^{\mathsf{len}_0,\mathsf{sign}_0} \right], \tag{2}$$

then there exists a constant $c \geq 1$, such that $\tau^* - \tau_0 > \frac{1}{k^c}$. Recall that the tuple length $\mathsf{len}_0 < t$. However, since MM_0 is in $\mathbb{D}_{N,t}$, we know that there exists another label ℓ_1 in MM_0 with tuple length t. We now define another multi-map MM_1 in $\mathbb{D}_{N,t}$ with signature sign_1 where we swap the tuple lengths of ℓ_0 and ℓ_1 such that ℓ_0 now has tuple length t and ℓ_1 has tuple length len_0. By definition,

$$E\left[X^{t,\mathsf{sign}_1} \right] \geq \tau^*. \tag{3}$$

From Equations (2) and (3), we have:

$$E\left[X^{t,\mathsf{sign}_1} \right] - E\left[X^{\mathsf{len}_0,\mathsf{sign}_0} \right] \geq \tau^* - \tau_0.$$

[7] We note that if the expected length of all encrypted responses are at most negligibly smaller than τ^* then they are $\Omega(\tau^*)$ by definition.

[8] Assuming that the length of the largest encrypted sub-response on any server $\mathbf{S}_i$ over all $\mathsf{MM} \in \mathbb{D}_{N,t}$ is N. This is a loose assumption, since N is the total number of values in the input multi-map. The theorem holds for any finite upper bound on the length of the encrypted sub-responses.

From the definition of the total expected response length, we have the following:

$$\sum_{i=1}^{n} E\left[X_i^{t,\mathsf{sign}_1} \right] - \sum_{i=1}^{n} E\left[X_i^{\mathsf{len}_0,\mathsf{sign}_0} \right] \geq \tau^* - \tau_0.$$

Then there exists a server $\mathbf{S}_j$ of the n servers where:

$$E\left[X_j^{t,\mathsf{sign}_1} \right] - E\left[X_j^{\mathsf{len}_0,\mathsf{sign}_0} \right] \geq \frac{\tau^* - \tau_0}{n}.$$

Writing the expectation as a sum over $(N+1)$ possible sub-response lengths, each at most N, we know there also exists some value x^*, $0 \leq x^* \leq N$, such that:

$$\Pr\left[X_j^{t,\mathsf{sign}_1} = x^* \right] - \Pr\left[X_j^{\mathsf{len}_0,\mathsf{sign}_0} = x^* \right] \geq \frac{\tau^* - \tau_0}{n \cdot N \cdot (N+1)}. \tag{4}$$

Now, notice that the value of the random variable $X_j^{\mathsf{len},\mathsf{sign}}$ is the length of the encrypted sub-response on server $\mathbf{S}_j$. Recall that the length of the encrypted response is *distinct* from the true response length (or volume); can include padding, dummies and overlaps; and is *always included in the view* of a corrupted server. Also notice that by our earlier assumption, the fraction on the right-hand-side of Equation (4) is non-negligible in k.

Then, given Equation (4), we construct an adversary $\mathcal{A}$ who corrupts server $\mathbf{S}_j$ and plays the multi-server volume-hiding game for the leakage profile $\mathcal{L}_\Sigma$:

- generate the signatures sign_0 and sign_1 as above and send them to the challenger $\mathcal{C}$,
- receive the setup leakage $\mathcal{L}_{\mathsf{Setup}}(k, \mathsf{MM}_\eta)$,
- choose the label ℓ_0 for the query sequence and receive the query leakage $\mathcal{L}_{\mathsf{Query}}(k, \mathsf{MM}_\eta, \ell_0)$,
- run the simulator of the scheme Σ_{MM} using the leakage $\mathcal{L}_\Sigma$ to generate the view of the corrupted server $\mathbf{S}_j$,
- if the length of the encrypted sub-response on $\mathbf{S}_j$ is x^*, output 1, else 0.

By the security of Σ_{MM}, the simulated view of $\mathbf{S}_j$ must also satisfy Equation (4). Then, from the definition of the volume-hiding game:

$$p_{1,j}^{\mathcal{A},\mathcal{L}_\Sigma}(N,n,t) - p_{0,j}^{\mathcal{A},\mathcal{L}_\Sigma}(N,n,t) \geq \frac{\tau^* - \tau_0}{n \cdot N \cdot (N+1)} > 0,$$

which violates the definition of a multi-server volume-hiding leakage profile.

■

If the scheme Σ_{MM} is *correct*, then $\tau^* \geq t$ over all multi-maps, since the scheme must retrieve at least the length of the original input tuple. This gives us the following bound as a direct corollary of Theorem 1.

Let $F\colon \{0,1\}^k \times \mathbb{L}_{\mathsf{MM}} \to [n]$ be a pseudorandom function. Then consider the following $\mathsf{Transform}$ and Split algorithms:

- $(\mathsf{TMM}, \mathsf{tmap}) \leftarrow \mathsf{Transform}(\mathsf{MM}, \mathsf{params} = \bot, n)$:
  1. initialize an empty map tmap;
  2. for every $\ell \in \mathbb{L}_{\mathsf{MM}}$, set $\mathsf{tmap}[\ell] := \ell$;
  3. output $(\mathsf{MM}, \mathsf{tmap})$.
- $(\mathsf{TMM}_1, \mathsf{TMM}_2, \ldots, \mathsf{TMM}_n, \mathsf{smap}) \leftarrow \mathsf{Split}(\mathsf{TMM}, \mathsf{tmap}, n)$:
  1. sample $K \xleftarrow{\$} \{0,1\}^k$;
  2. initialize n empty multi-maps $\mathsf{TMM}_1, \mathsf{TMM}_2, \ldots, \mathsf{TMM}_n$;
  3. initialize an empty map smap;
  4. for each $\ell \in \mathbb{L}_{\mathsf{MM}}$:
     (a) compute $i := \mathsf{F}_K(\ell)$;
     (b) set $\mathsf{TMM}_i[\ell] := \mathsf{MM}[\ell]$;
     (c) set $\mathsf{smap}[\ell] := \mathbf{S}_i$.
  5. output $(\mathsf{TMM}_1, \mathsf{TMM}_2, \ldots, \mathsf{TMM}_n, \mathsf{smap})$.

Fig. 2. VST: The Vertical Shard Transform-and-Split

Corollary 1. *Any* correct *multi-server encrypted multi-map scheme Σ_{MM} in the transform-and-split model with a volume-hiding leakage profile $\mathcal{L}_\Sigma$, must have expected communication complexity $\Omega(t)$, for any $n \geq 1$, and for all $N \geq t \geq 1$, for any C such that $\#\mathrm{C} \geq 1$, for all input multi-maps $\mathsf{MM} \in \mathbb{D}_{N,t}$, where the expectation is computed over the randomness of the* $\mathsf{Transform}$ *and* Split *algorithms.*

Note that our lower bound is in a fairly restrictive setting: a static encrypted multi-map scheme, a semi-honest adversary, one server corruption, and one query in the volume-hiding game. It is therefore unlikely that schemes with stronger security (e.g. zero-leakage [30], subliminal [8]), supporting dynamic volume-hiding (e.g. [4]), or stronger adversary models can get around our bound.

7 VSM: Vertical Sharding with Optimal Communication

In this section, we briefly discuss our transform-and-split encrypted multi-map scheme VSM which has optimal communication at the cost of partial volume leakage. Our scheme makes use of the *vertical shard* $\mathsf{Transform}$ and Split algorithms shown in Figure 2. In essence, VST is to encrypted multi-maps what hash-based sharding is to plaintext databases: a simple, key-driven partition that disperses information horizontally, thereby confining volume leakage to narrow vertical slices while preserving the efficiency of direct label look-ups.

Overview of VST. The Vertical Shard Transform-and-Split algorithms randomly assign the tuples of the input multi-map to the n servers. In other words, $\mathsf{Transform}$ outputs a transformed multi-map TMM that is exactly the same as the input multi-map MM; and Split uses a pseudorandom function F_K to split the labels across the n servers. When combined with an appropriate (single-server)

encrypted multi-map scheme Σ_{MM} as in our framework (Figure 1), the VST gives rise to different multi-server encrypted multi-map schemes.

VSM*: A Vertical Shard Multi-map Scheme.* VSM uses the VST in conjunction with a standard leaky encrypted multi-map scheme Σ_{MM}, such as Π_{bas} [15]. The storage and communication complexities of VSM are optimal, since we only incur the overheads for encryption itself. Since the standard Σ_{MM} has the leakage profile $(\mathsf{patt}_{\mathsf{S}}, \mathsf{patt}_{\mathsf{Q}}) = (\mathsf{dsize}, (\mathsf{qeq}, \mathsf{rlen}))$, the resulting multi-server scheme reveals to a corrupted server the total number of values assigned to it, and the individual volumes of each query assigned to it. We provide a more detailed description, the formal leakage profile Λ_{VSM}, and the security proof in the full version.

References

1. Agarwal, A., Espiritu, Z.: Sequentially consistent concurrent encrypted multimaps. In: 2025 IEEE 10th European Symposium on Security and Privacy (EuroSP). pp. 631–654 (2025). https://doi.org/10.1109/EuroSP63326.2025.00042
2. Agarwal, A., Kamara, S.: Encrypted distributed dictionaries. Cryptology ePrint Archive, Paper 2019/1126 (2019), https://eprint.iacr.org/2019/1126
3. Agarwal, A., Kamara, S., Moataz, T.: Concurrent encrypted multimaps. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024. pp. 169–201. Springer Nature Singapore, Singapore (2025)
4. Amjad, G., Patel, S., Persiano, G., Yeo, K., Yung, M.: Dynamic volume-hiding encrypted multi-maps with applications to searchable encryption. Proc. Priv. Enhancing Technol. **2023**(1), 417–436 (2023). https://doi.org/10.56553/popets-2023-0025, https://doi.org/10.56553/popets-2023-0025
5. Ando, M., George, M.: On the Cost of Suppressing Volume for Encrypted Multi-maps. In: Proceedings on Privacy Enhancing Technologies. vol. 2022, pp. 44–65. De Gruyter (2022)
6. Asharov, G., Naor, M., Segev, G., Shahaf, I.: Searchable symmetric encryption: optimal locality in linear space via two-dimensional balanced allocations. In: Wichs, D., Mansour, Y. (eds.) Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016. pp. 1101–1114. ACM (2016). https://doi.org/10.1145/2897518.2897562, https://doi.org/10.1145/2897518.2897562
7. Asharov, G., Segev, G., Shahaf, I.: Tight tradeoffs in searchable symmetric encryption. In: Annual International Cryptology Conference. pp. 407–436. Springer (2018)
8. Bahi, A., Kamara, S., Moataz, T., Noubir, G.: Subliminal encrypted multi-maps and black-box leakage absorption. Cryptology ePrint Archive, Paper 2024/1708 (2024), https://eprint.iacr.org/2024/1708
9. Bienstock, A., Patel, S., Seo, J.Y., Yeo, K.: Near-Optimal oblivious Key-Value stores for efficient PSI, PSU and Volume-Hiding Multi-Maps. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 301–318. USENIX Association, Anaheim, CA (Aug 2023), https://www.usenix.org/conference/usenixsecurity23/presentation/bienstock
10. Blackstone, L., Kamara, S., Moataz, T.: Revisiting leakage abuse attacks. In: Network and Distributed System Security Symposium (NDSS '20) (2020)

11. Bost, R.: Sophos - forward secure searchable encryption. In: ACM Conference on Computer and Communications Security (CCS '16) (20016)
12. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: ACM Conference on Communications and Computer Security (CCS '15). pp. 668–679. ACM (2015)
13. Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Advances in Cryptology - CRYPTO '13. Springer (2013)
14. Cash, D., Tessaro, S.: The locality of searchable symmetric encryption. In: Advances in Cryptology - EUROCRYPT 2014 (2014)
15. Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. In: Network and Distributed System Security Symposium (NDSS '14) (2014)
16. Cash, D., Ng, R., Rivkin, A.: Improved structured encryption for SQL databases via hybrid indexing. In: Sako, K., Tippenhauer, N.O. (eds.) Applied Cryptography and Network Security: 19th International Conference, ACNS 2021, Kamakura, Japan, June 21-24, 2021, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12727, pp. 480–510. Springer (2021). https://doi.org/10.1007/978-3-030-78375-4_19
17. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS) **26**(2), 4 (2008)
18. Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Advances in Cryptology - ASIACRYPT '10. Lecture Notes in Computer Science, vol. 6477, pp. 577–594. Springer (2010)
19. Cohen, W.W.: Enron email dataset (2015), https://www.cs.cmu.edu/~enron/, collected by the CALO Project. May 7, 2015 version. Last modified May 8, 2015
20. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. In: ACM Conference on Computer and Communications Security (CCS '06). pp. 79–88. ACM (2006)
21. Garimella, G., Pinkas, B., Rosulek, M., Trieu, N., Yanai, A.: Oblivious key-value stores and amplification for private set intersection. In: Advances in Cryptology – CRYPTO 2021. Lecture Notes in Computer Science, vol. 12826, pp. 395–425. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84245-1_14
22. Goh, E.J.: Secure indexes. Tech. Rep. 2003/216, IACR ePrint Cryptography Archive (2003), See http://eprint.iacr.org/2003/216
23. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM **43**(3), 431–473 (1996)
24. Grubbs, P., Lacharité, M., Minaud, B., Paterson, K.G.: Pump up the volume: Practical database reconstruction from volume leakage on range queries. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 315–331. ACM (2018). https://doi.org/10.1145/3243734.3243864, https://doi.org/10.1145/3243734.3243864
25. Islam, M., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: NDSS (2012)
26. Kamara, S., Moataz, T.: Boolean searchable symmetric encryption with worst-case sub-linear complexity. In: Advances in Cryptology - EUROCRYPT '17 (2017)
27. Kamara, S., Moataz, T.: SQL on Structurally-Encrypted Data. In: Asiacrypt (2018)

28. Kamara, S., Moataz, T.: Computationally volume-hiding structured encryption. In: Advances in Cryptology - Eurocrypt' 19 (2019)
29. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: ACM Conference on Computer and Communications Security (CCS '12). ACM Press (2012)
30. Kamara, S., Moataz, T., Ohrimenko, O.: Structured encryption and leakae suppression. In: Advances in Cryptology - CRYPTO '18 (2018)
31. Kellaris, G., Kollios, G., Nissim, K., Neill, A.O.: Generic attacks on secure outsourced databases. In: ACM Conference on Computer and Communications Security (CCS '16) (2016)
32. Larsen, K.G., Simkin, M., Yeo, K.: Lower bounds for multi-server oblivious rams. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography. pp. 486–503. Springer International Publishing, Cham (2020)
33. Meng, X., Kamara, S., Nissim, K., Kollios, G.: Grecs: Graph encryption for approximate shortest distance queries. In: ACM Conference on Computer and Communications Security (CCS 15) (2015)
34. MongoDB: https://www.mongodb.com/. mongodb: The world's leading modern database, https://www.mongodb.com/
35. Patel, S., Persiano, G., Yeo, K.: Lower bounds for encrypted multi-maps and searchable encryption in the leakage cell probe model. Springer-Verlag (2020). https://doi.org/10.1007/978-3-030-56784-2_15
36. Patel, S., Persiano, G., Yeo, K., Yung, M.: Mitigating leakage in secure cloud-hosted data structures: volume-hiding for multi-maps via hashing. In: Conference on Computer and Communications Security (CCS '19). pp. 79–93 (2019)
37. Persiano, G., Yeo, K.: Lower bound framework for differentially private and oblivious data structures. In: Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I. pp. 487–517 (2023). https://doi.org/10.1007/978-3-031-30545-0_17, https://doi.org/10.1007/978-3-031-30545-0_17
38. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data. In: IEEE Symposium on Research in Security and Privacy. pp. 44–55. IEEE Computer Society (2000)
39. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: Network and Distributed System Security Symposium (NDSS '14) (2014)
40. Wang, J., Sun, S.F., Li, T., Qi, S., Chen, X.: Practical volume-hiding encrypted multi-maps with optimal overhead and beyond. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 2825–2839. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3559345, https://doi.org/10.1145/3548606.3559345
41. Yeo, K.: Lower bounds for (batch) pir with private preprocessing. In: Ventura, L. (ed.) Advances in Cryptology – EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23–27, 2023, Proceedings, Part I, pp. 535–564. Springer-Verlag (2023). https://doi.org/10.1007/978-3-031-30545-0_18, https://dl.acm.org/doi/10.1007/978-3-031-30545-0_18
42. Zhao, Z., Kamara, S., Moataz, T., Zdonik, S.: Encrypted databases: From theory to systems. In: Proceedings of the 11th Conference on Innovative Data Systems Research (CIDR) (2021), https://cidrdb.org