Encrypted Databases: New Volume Attacks against Range Queries *

Zichen Gui[1], Oliver Johnson[1], and Bogdan Warinschi[1,2]

[1]University of Bristol
[2]DFINITY
{zg13988,o.johnson,csxbw}@bristol.ac.uk

Abstract

We present a range of novel attacks which exploit information about the *volume* of answers to range queries in encrypted database.

Our attacks rely on a strategy which is simple yet robust and effective. We illustrate the robustness of our strategy in a number of ways. We show how i) to adapt the attack for several variations of a basic usage scenario ii) to defeat countermeasures intended to thwart the premise of our basic attack and iii) to perform partial reconstruction of secret data when unique reconstruction is information theoretically impossible. Furthermore, over the state of the art, our attacks require one order of magnitude fewer queries. We show how to improve the attacks even further, under the assumption that some partial information is known to the adversary.

We validate experimentally all of our attacks through extensive experiments on real-world medical data and justify theoretically the effectiveness of our strategy for the basic attack scenario. Our new attacks further underscore the difficulty of striking an appropriate functionality-security trade-off for encrypted databases.

1 Introduction

The idea of outsourcing data storing and processing to an untrusted server has triggered significant cryptographic research on methods to hide the information both from eavesdroppers but also from the cloud itself [8, 4, 19, 3, 2, 9, 13, 7]. In typical solutions, data is stored encrypted, but in a way which allows the server to process (encrypted) queries to produce an encrypted response which only the client can decrypt.

Unfortunately, a satisfactory trade-off between efficiency and security proves elusive. All schemes inherently leak various types of information which can be

*The paper will appear in ACM CCS 2019.

exploited by the adversary, sometime with devastating effects. Early designs [18, 8] allow the server to learn the access pattern induced by queries. For each query the adversary learns which records are accessed when it processes each query of the client. It recently became quite clear that the access pattern can help to recover all of the private data [15, 17, 11].

More recently, a new attack vector which relies on significantly less information had been identified [15, 12]. Here, the adversary disregards completely the internal structure of the answers to queries, and only looks at their *size*. This setting models, for instance, an honest but curious server who learns the number of encrypted record which corresponds to an encrypted query. It also models an eavesdropper who observes the traffic between a client and a user over some secure channel. Indeed, even if the communication between a user and a server is via modern secure channels (e.g. as implemented by TLS 1.3) a passive eavesdropper may learn the traffic size information since such channels do not hide the size of transmitted messages.

1.1 Volume leakage attacks against encrypted databases.

Range queries in encrypted databases. We continue a recent line of work which shows how to exploit volume information in encrypted databases which support range queries. In this setting database records are associated with some numerical label $i \in \{1, 2, \ldots, N\}$ for some N. For example, in medical databases these labels can record a patient's age, the number of days the patient has been in the hospital, the severity of the illness, etc. If v_i is the number of records associated with value i then the vector $\mathbf{v} = (v_1, v_2, \ldots, v_N)$ describes the database counts for the individual labels. We will refer to the entries in $\mathbf{v}$ as *elementary volumes*. In this paper we consider attackers which try to recover (information about) $\mathbf{v}$. This information can be highly sensitive. For example, if labels record various diseases in a regional hospital, the counts reflect the incidence of that disease in the population served by that hospital. This type of information can be (ab)used by insurance companies to reduce the risk they take at the expense of policy holders.

Attacker model. The attack model we consider in this paper assumes an adversary who obtains volume information related to range queries. These are queries of the form $Query(i, j)$. The answer consists of all records with labels between i and j, so the volume of the answer is $v_i + \ldots + v_j$. We will refer to $j - i + 1$ as the the window size of the query. The adversary learns (potentially noisy) information about the volume of the response and, we emphasize that, the adversary does not learn information about the query itself.

The following notation is useful to make more precise the setting which we consider. For fixed database (counts) $\mathbf{v}$ and fixed set of queries $\mathbf{q}$ the adversary learns some function $\mathcal{L}(\mathbf{v}, \mathbf{q})$. This may comprise precisely the volumes which corresponds to the queries in $\mathbf{q}$, but can also include noisy information whenever noise is injected in the communication between the parties. In an attack, both $\mathbf{v}$ and $\mathbf{q}$ are sampled according to some distributions $\mathcal{V}$ and $\mathcal{Q}$; the adversary

is given $\mathcal{L}(\mathbf{v}, \mathbf{q})$ and returns $\mathbf{o}$ some which represents the information it had learned. Various generalizations are possible. For example, one may decide to include in the input of the adversary partial information about $\mathbf{v}$, or have the distribution $\mathcal{V}$, $\mathcal{Q}$ sampled from some distributions themselves (to indicate that the adversary may not have full knowledge about these).

The success of the adversary can then be measured in how "close" are $\mathbf{o}$ and $\mathbf{v}$, for some appropriately chosen notion of closeness. For example, for most of our attacks we look at the probability that $\mathbf{o} = \mathbf{v}$, which corresponds to full reconstruction of the database counts.

Prior work. To date, there are two known volume leakage attacks. The breakthrough KKNO attack due to Kellaris et al. [15] shows that after observing $\Theta(N^4 \log N)$ queries distributed independently and uniformly at random (over the set of all possible queries) an adversary can recover all of the database counts. The attack reduces the problem of recovering database counts to factoring polynomials. The attack is rather inefficient and is more of a proof of concept. Recently, Grubbs et al. [12] have shown an attack (henceforth the GLMP attack) with significantly reduced query complexity. Their attack works for an adversary who sees the answers to *all* of the $\Theta(N^2)$ possible queries. A standard argument shows that this event occurs with overwhelming probability after $\Theta(N^2 \log N)$ uniformly independent queries. The GLMP attack reduces the recovery problem to that of clique finding in an appropriately defined graph. Since this is an NP-complete problem, the GLMP attack employs heuristics to deal with the inherent computational complexity of the problem.

Before we move on, we briefly discuss two aspects of the above attacks. Although they expose a clear source of insecurity the attacks rely on strong assumptions on the information the adversary learns. Indeed, since at the moment we have little understanding of what is the query distribution in practical applications, it is difficult to confirm the assumptions required by the attacks. Is it unclear how reasonable it is to assume that the adversary gets to observe answers to all possible queries. The independence and uniformity assumptions on the query distribution are perhaps even harder to justify.

Furthermore, it seems easy to defend against these attacks since they do not seem robust under small departures from the assumptions on which they rely. Put differently, do the attacks fail if a few queries are missing? What if the adversary never gets to see the answer to $Query(1, N)$ or queries of the form $Query(i, i)$, both of which are crucial for the GLMP attack to work?

These are not merely rhetorical questions. One tempting proposition is to deploy protocols with a modicum of protection which invalidate the assumptions required by the attacks. For example, one could simply ensure that elementary volumes are never returned, prevent queries for large ranges, or introduce noise in the communication. Each of these countermeasures thwarts existing attacks, with no obvious way to bypass them.

1.2 Our results

We present novel volume leakage attacks which deal with all of the scenarios outlined above. Our attacks use a simple yet effective strategy which takes full advantage of the information present in volume leakage associated with range queries. Over the state of the art, our attacks require one order of magnitude fewer queries and are more versatile and robust: they tolerate missing and spurious queries, and can be easily adapted for settings where noise has been added to the information leaked. Our attacks share a robust strategy where we use the leaked information iteratively build larger solutions from smaller ones: different leaked information imposes constraints on valid solutions which helps to keep the process efficient. We validate our attacks both theoretically and experimentally. Below we describe our results in more details. We start with a basic scenario which is useful to understand our basic attack strategy and incrementally build towards our more involved attacks which themselves require additional insights.

Basic attack. Our basic strategy is illustrated best with a basic scenario where the adversary sees all queries for ranges with window size up to some upper bound b (setting $b = N$ yields the setting of [12]). Here, and for the rest of the attacks which we consider, we make no specific assumption on the query distribution and do not specify $\mathbf{q}$ either; the attack works for any distribution which allows the adversary to learn this information.

Formally, if $\mathbf{v} = (v_1, \ldots, v_N)$ is the vector of database counts and b is a bound on the query window size, in the basic setting the information learned by the adversary is described by the function

$$\mathcal{L}_b(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i \mid \ y - x + 1 \leq b \right\}.$$

The problem of complete recovery of database counts is: Given some volume leakage W, find the solution(s) to the equation

$$W = \mathcal{L}_b(\mathbf{v}). \tag{1}$$

We describe an algorithm which recovers all solutions to the above equation. Our algorithm constructs solutions to the equation, incrementally. At any given point, we maintain a set of valid *partial* solutions of the form $\mathbf{u} = (w_1, w_2, \ldots, w_k)$. Each such partial solution satisfies the invariant $\mathcal{L}_b(\mathbf{u}) \subseteq W$. Every contiguous subarray of $\mathbf{v}$ satisfies the property and, moreover, this simple formulation encapsulates a large number of simultaneous constraints. At every step, for each partial solution $\mathbf{u}$ in turn, we attempt to extend it with some volume $w \in W$ subject to the condition above. If for all $w \in W$ the extension fails, then we discard $\mathbf{u}$; otherwise we add the result to the set of solutions of length $k + 1$. We establish both theoretically and experimentally that this greedy strategy is highly effective: W is highly structured and allows us to quickly eliminate partial solutions which are not subarrays of $\mathbf{v}$.

Experimental validation. We experimentally validate our analysis in two different ways. First, we use data sampled according to the distributions for which we provide theoretical bounds. Here, we show that either the theoretical bound matches quite well the experimental results, or that it is in fact overly pessimistic. Studying the effectiveness of our attack on this type of synthetic data also helps to identify causes due to which our attack fails.

More interestingly, we run the attack against real world data. Full details of the data we use are in Appendix A. Here we provide a brief overview. We use the national Inpatient Sample database from the Healthcare Cost and Utilization Project (HCUP) dataset [10] from three years, which provides more than three thousand databases, one per hospital. Our attacks recover volume count information for the attributes admission month (`AMONTH`), number of diagnoses (`NDX`), number of procedures (`NPR`), age (`AGE`), and length of stay (`LOS`).

For our basic attack, both the analytic and experimental work show that our attacks succeed for rather small b. For each attribute in turn, we run our algorithm (for varying leakage parameters b) and record the fraction of databases for which we recover *precisely* the underlying database, up to reflection. For some attributes with small N, like `NDX` and `NPR`, where N is either 16 or 26 we can recover database counts for over 80% of the databases, even if the adversary only sees queries with window size b as little as 3. Attribute `AGE` (where $N = 83$) is harder to recover; yet as soon as the adversary sees queries of size up to $b = 8$ we can efficiently recover more than one in three databases uniquely (NB: for simplicity we will often say we recover the database rather than database counts). Our attack requires $\Theta(N)$ queries, an order of magnitude improvement over the best attack of Grubbs et al. [12] and works in seconds on a commodity laptop (See Appendix A for a brief comparison.)

Robustness against variations of the leaked information. It is easy to identify scenarios where the communication between the client and the server contains information that does not fit our basic attack scenario. For example, the client may issue queries with windows size larger than b or the communication protocol may actively try to modify the leaked information. Countermeasures which have been suggested in the literature include suppressing answers to some queries, adding fake queries and answers to the communication and adding noise to the answer themselves. We show that our strategy still works as is, or can be adapted to deal with such variations.

For example, we consider a setting where the leakage is noisy and comprises additional spurious volumes. In the first instance we look at the case where the spurious volumes correspond to valid queries but with larger window size than expected. Although, as expected, we notice a degradation in the success of our attacks, they still perform exceedingly well. For example, we experiment with an adversary which attempts to recover the `NPR-26` attribute counts in a setting where for every two true query we add a randomly chosen query of larger size. Here the success of our adversary drops from around 80% to around 40% which is still devastating.

Interestingly, the attacks work even better if the noise added is selected

uniformly at random (e.g. as an explicit attempt to hide volume information). Here, we recover more than 60% of the databases for the same attribute (even when for each true query we add a fake one). This indicates that random noise is a poor defense which can be easily filtered out using the relations present in the genuine part of the leakage.

A potentially more promising countermeasure is to pad answers with fake records. Although this countermeasure perturbs the structure present in the leakage we observe that approximate variants of the type of constraints we use in our basic attack still hold. As a proof of concept, we show how adapt our attack to break this type of defense when Z is a discrete uniform random distribution. Here we give a complete recovery algorithm which works for a specific setting of the padding parameters. Even for better chosen parameters we show that our attack can be adapted to perform approximate reconstruction of the database counts.

Finally, we also show how to deal with the case where some of the queries in the expected set of queries are missing.

Bounded window size. All existing attacks, including those introduced above, are for settings where the adversary sees all elementary volumes. A natural countermeasure is to prevent direct information about elementary volumes from being revealed to the attacker. For example, one can demand that the range of queries is at least 2 or, as suggested by Grubbs et al [12], to batch the answer to several queries into a single answer. In this case, the leakage simply does not contain any elementary volume and it is not clear if these can be recovered.

We extend our attacks to a setting where the window size of queries is bounded from above *and* from below, that is the volumes of queries with small and large window sizes are hidden. We model the resulting leakage as a function

$$\mathcal{L}_{a,b}(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i \mid \ a \leq y - x + 1 \leq b \right\}$$

parameterized by bounds a and b.

To cope with the new setting which is quite different from the one in our basic attack we require a different way of using the information present in the leakage. Without going too much into detail, we present a two-stage attack where our basic attack plays a key role. In the first stage we regard volumes of ranges of size a as elementary and apply our basic attack. We obtain as a result an ordered set of volumes which, with high probability, should correspond to consecutive ranges of size a. In the second stage we obtain a different ordered set of volumes, again by using our basic algorithm but with additional constraints which use the output of the first stage. With some care, we can then recover $\mathbf{v}$ from these two pieces of information.

Although $\mathcal{L}_{a,b}$ leaks significantly less information than in our basic scenario, our attack shows that there is still enough structure in W. Indeed, it turns out that the HCUP databases are still extremely vulnerable to our modified attack. We run our attack trying to recover the attributes `NDX-26` and `NPR-26` with

different a and b. In the worst case, where $a = 3$ and $b = 9$ we can still obtain the precise counts for 78% of the databases for the attribute `NDX-26`.

Partial reconstruction attacks. In the above discussion and analysis we consider, quite conservatively, that the adversary wins only when it manages to successfully reconstruct the volume counts (up to reflection). While the level of insecurity we demonstrate is already rather devastating, we push our attack even further.

Notice that our attack may fail in two distinct ways. First, the attack aborts when the number of partial solutions it maintains grows beyond a certain bound. One obvious way to extend the attack is to simply increase the space allowance of the algorithm (currently set to 1GB).

More interestingly, we take a closer look at the case when there is insufficient information in W to uniquely recover $\mathbf{v}$, in which case the attacks return multiple solutions to the equation $\mathcal{L}_b(\mathbf{v}) = W$. This is the case, for example, when $\mathbf{v}$ contains a subarray which can be permuted without changing the leakage. In this case, the solutions found by our algorithm still overlap on a significant proportion of labels. In effect, this means that our attack reconstructs the (unambiguous) part of the database. Clearly, even if the privacy breach is not total, the leak should be considered problematic.

This observation motivates a more refined measure of adversarial success where its goal is to only partially reconstruct the volume counts. Of the several different choices on how to quantify the success of the adversary we consider two. The first simply counts the fraction of labels for which the attack recovers the correct counts; the second, also accounts for the size of the counts themselves. For example, an adversary may correctly identify counts for a small fraction of the labels, but the total number of records associated with these labels can be overwhelming.

We experiment with attacks against the length of stay attribute (which by nature leads to leakage with a large set of valid preimages). Here, our attack aborts for roughly 15%. However, for the remaining databases it recovers correctly the value associated with more than 95% of the records (divided across at least 37% of the labels).

Partial information. Another way to deal boost the success of our attacks it to somehow incorporate side information about and we briefly explore two distinct options. The first uses partial information to disambiguate between multiple valid solutions. Here, we assume that the adversary has access to a very small section of the real database (e.g. learns the volume of a single query) and filters out solutions which are not consistent with the adversary's view. Second, we also incorporate side information in the attack more directly. More precisely, we assume that the adversary knows the "shape" of the distribution (e.g. that it is decreasing) and use this information to prune the search. In both cases, we confirm the intuition that side-information can significantly improve the success of the our attacks.

1.3 Discussion

Assumptions. We assume the adversary is able to collect *all* queries within some bounded window size and assume that the database is dense, meaning that $v_i \neq 0$ for all $i = 1, \ldots, N$. This is an assumption that prior work also makes. However, we explain later in the paper that our strategy can be adapted to work if the database is close to being dense.

We also make the reasonable assumption that the maximum label N is known by the adversary. For example, in medical records the age of patients belong to some standard set (i.e. age groups), the diagnostics are divided across some canonical groups, and an address filed may correspond to known postal codes.

Significance. Our attacks work under strictly weaker assumptions than those required for existing attacks, but their immediate impact on deployed systems is difficult to quantify. The problem, shared with all works in this space, is that the research community does not have a good handle on what are realistic query distributions for range queries. To compensate for this lack of understanding, we avoid to make specific assumptions on the distribution of queries. Instead we focus on what one can interpret as being worst case scenarios for security where we characterize the information which we can exploit (independent of how it can be obtained by the adversary). Furthermore, while we are not (and cannot be!) exhaustive we explore multiple leakage scenarios all inspired by natural uses of the scheme and by the effect of countermeasures some actually suggested in the literature.

The robustness of our basic strategy under all of these variations serves as evidence that volume leakage for range queries is highly damaging and requires novel countermeasures to be contained.

2 Basic Attack

In this section, we describe and analyse our basic reconstruction attack. We assume that only queries of small windows are observed by the attacker. More formally, we define the leakage function as

$$\mathcal{L}_b(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i \mid \ y - x + 1 \leq b \right\}. \tag{2}$$

The adversary then learns $W = \mathcal{L}_b(\mathbf{v})$ for some database $\mathbf{v} = (v_1, \ldots, v_N)$; his goal is to recover $\mathbf{v}$. By setting $b = N$, we recover the leakage function in the previous attacks [15, 12]. It is the only case for which previous attacks work: both attacks require volumes generated by *all* possible queries. On the other hand, our attack works with considerably smaller b.

2.1 Reconstruction Algorithm

Intuition. Given a tuple of observed volumes, say $(w_1, \ldots, w_k)$, we can tell if it looks like a subarray of $\mathbf{v}$, since we know a set of conditions need to be satisfied: all the sums of two consecutive volumes need to be present in W; similarly, the sums of three to b consecutive volumes need to be present in W. Given a large number of constraints, it is unlikely that an arbitrary choice of $(w_1, \ldots, w_k)$ meets all the constraints. Hence, a partial solution of length k is likely to be a continuous segment of $\mathbf{v}$.

Our basic attack is a variant of breadth-first search with pruning which extends partial solutions iteratively. There are several choices for the initial partial solutions. For certain databases, the minimum observed volume suffices as an initial partial solution. This choice may not work on some databases as the number of partial solutions grows too quickly. In that case, we opt for a clique-finding subroutine inspired by GLMP [12] to generate partial solutions of length b. In each iteration, our attack tries to extend the partial solutions found in the previous iteration to the left and right by checking the new constraints introduced by the new volume. Partial solutions that cannot be extended by any observed volume are discarded. After obtaining the partial solutions of length N, we check if the solutions generate the expected leakage, and the solutions that do not are discarded. The remaining solutions are returned by our attack. Pseudocode of our attack is shown in Algorithm 2.

Initial Solution via Clique-finding. An obvious starting partial solution is the smallest observed volume: since all elementary volumes are observed the smallest volume is necessarily an elementary volume. Looking ahead, this strategy may not work (e.g. when the smallest volume is suppressed). We therefore employ a more robust mechanism inspired by the GLMP attack and which shares some ideas with a heuristic which complements the KKNO attack. Below, we describe how an initial solution can be identified by finding a clique in a certain graph. Let W be the set of observed volumes. We define the set of complemented volumes C to be $\{v \mid v \in W \wedge \max(W) - v \in W\} \cup \{\max(W)\}$. Nodes of our graph are defined by the elements of C. For $v_1, v_2 \in C$, there is an edge between nodes v_1 and v_2 if $|v_1 - v_2| \in W$. We know $\max(W)$ is the sum of b elementary volumes, so a clique of size b with one of the volumes as $\max(W)$ describes a partial solution of length b.

In GLMP the reconstruction of the entire database is reduced to finding a clique (with N nodes). Here we only use clique finding to initialize our search, so we only need to recover a small segment of size b of the database.

Detailed pseudocode of our clique-finding algorithm can be found in Algorithm 1. In line 13, we use $g\langle i \rangle$ to mean i-th smallest volume in g. It is also worth pointing out that our algorithm employs a similar pruning strategy suggested in Appendix E of the KKNO paper [15]. Without going into too many details, that algorithm verifies similar constraints with our pruning mechanism to identify a plausible segment of the hidden database.

Attack Overview. Pseudocode of our basic attack is shown in Algorithm 2.

Algorithm 1 Finding initial solution set

1: **input** $W = \{\sum_{i=x}^{y} v_i \mid y - x + 1 \leq b\}, b$
2: **output** $\{(w_1, \ldots, w_b) \mid w \in W \wedge \mathcal{L}_b(w_1, \ldots, w_b) \subset W \wedge \sum_i w_i = \max(W)\}$

3: **procedure** INITIAL SOLUTION(W, b)
4: $C = \{v \mid v \in W, \max(W) - v \in W\} \cup \{\max(W)\}$
5: $G_2 = \{\{v, \max(W)\} \mid v \in C, v \neq \max(W)\}$
6: **for** $i \leftarrow 3, b$ **do**
7: $G_i = \{\}$
8: **for** $g \in G_{i-1}, v \in C$ **do**
9: **if** $\{|h - v| \mid h \in g\} \subset W$ **then**
10: $G_i = G_i \cup \{g \cup \{v\}\}$
11: $\mathcal{S} = \{\}$
12: **for** $g \in G_b$ **do**
13: $s = (g\langle 1\rangle, g\langle 2\rangle - g\langle 1\rangle, \ldots, g\langle b\rangle - g\langle b-1\rangle)$
14: $\mathcal{S} = \mathcal{S} \cup \{s\}$
15: **return** $\mathcal{S}$

Our clique-finding algorithm (Algorithm 1) is used as the initial solution finding algorithm but other choices are possible. In line 4, the set of initial solutions are computed. The partial solutions are then iteratively extended by running procedures EXTEND LEFT and EXTEND RIGHT. In these two procedures, only the sums involving the new volume w has to be checked, as all the other constraints are checked in previous iterations. After obtaining the partial solutions of length N, we recompute the leakage associated to each one and discard those for which the result is different from the input to the algorithm.

2.2 Theoretical Analysis

CORRECTNESS. For convenience, we denote our basic attack algorithm as $\mathcal{A}_1$. It takes as input the set of observed volumes $W = \mathcal{L}_b(\mathbf{v})$, bound b and maximum label N. We say that $\mathcal{A}_1$ is correct if the output of the attack is precisely the set of databases that generates the given leakage. Formally, the correctness of our attack is established by the following theorem. The proof can be found in Appendix B.1.

Theorem 1 (Correctness of the basic attack)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* b *be any natural number less or equal to* N*. Let* $\mathcal{S}_N$ *be the output of* $\mathcal{A}_1$*, i.e.* $\mathcal{S}_N = \mathcal{A}_1(\mathcal{L}_b(\mathbf{v}), b, N)$*. Then*

1. $\forall \mathbf{v}' \in \mathcal{L}_b(\mathbf{v})^N, \mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v}) \Leftrightarrow \mathbf{v}' \in \mathcal{S}_N$,
2. $\mathbf{v} \in \mathcal{S}_N$.

The correctness property of the attack implies that our attack is optimal in terms of database recovery, meaning that there is no algorithm that can

Algorithm 2 Basic attack

input $W = \{\sum_{i=x}^{y} v_i \mid y - x + 1 \leq b\}, b, N$
output $\{(w_1, \ldots, w_N) \mid w_i \in W\}$

procedure ATTACK(W, b, N)
    $\mathcal{S}_b = $ INITIAL SOLUTION(W, b)
    for $i \leftarrow b + 1, N$ **do**
        $\mathcal{S}_i \leftarrow$ EXTENDLEFT($\mathcal{S}_{i-1}, W, b$)$\cup$
          EXTENDRIGHT($\mathcal{S}_{i-1}, W, b$)
    $\mathcal{S}_N \leftarrow \{s \mid s \in \mathcal{S}_N \wedge \mathcal{L}_b(\mathcal{S}_N) = W\}$
    return $\mathcal{S}_N$

procedure EXTEND LEFT($\mathcal{S}_i, W, b$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_1, \ldots, w_i) \in \mathcal{S}_i, w \in W$ **do**
        if $\{w + \sum_{j=1}^{k} w_j \mid k < b\} \subseteq W$ **then**
            $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w, w_1, \ldots, w_i)\}$
    return $\mathcal{S}'$

procedure EXTEND RIGHT($\mathcal{S}_i, W, b$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_1, \ldots, w_i) \in \mathcal{S}_i, w \in W$ **do**
        if $\{w + \sum_{j=1}^{k} w_{i-j+1} \mid k < b\} \subseteq W$ **then**
            $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w_1, \ldots, w_i, w)\}$
    return $\mathcal{S}'$

eliminate more partial solutions from the set of solutions, if there is no further information.

UNIQUENESS. Correctness of our algorithm only ensures that we recover all the valid solutions from the leakage. It is possible to have multiple databases (up to reflection) generating the same leakage. For example, $\mathcal{L}_3((1,1,1,2,1,1)) = \mathcal{L}_3((2,1,1,1,1,1)) = \{1,2,3,4\}$, so the databases $(1,1,1,2,1,1)$ and $(2,1,1,1,1,1)$ are indistinguishable given the leakage. Yet, our experiments show that our attacks recover unique solutions for an overwhelming amount of real-world databases. In this section we investigate why this is the case. We assume that the databases are randomly sampled from some underlying distribution $\tilde{V}$ and we are therefore interested in the probability that the database can be reconstructed uniquely from the leakage:

$$p_{unique} = \Pr_{\mathbf{v} \leftarrow \tilde{V}} \left[\mathcal{S} \leftarrow \mathcal{A}_1(\mathcal{L}_b(\mathbf{v}), b, N), \exists s, \mathcal{S} = \left\{ s, s^R \right\} \right].$$

It becomes immediately clear that the probability is computationally infeasible to compute from its analytical expression. We discuss how to use a series of

bounding and approximation techniques to find an estimation of the probability. We use a very rough approximation where we are interested in the event that all of the partial solutions discovered/maintained by our attack are *genuine*, that is they are solutions of the form $(v_i, \ldots, v_j)$ for some i and j or their reflections. First we bound the probability that at iteration b the set of partial solutions contains *only* genuine solutions. Here, we use the Chen-Stein method [6, 5] to estimate the distribution of the observed volumes and derive the probability that an arbitrary choice of b volumes from the set of observed volumes satisfies all the constraints. This allows us to derive the probability that all the partial solutions of length b are genuine solutions (Appendix B.2 Lemma 3). Then, we bound the probability that the solution set contains only genuine solutions at iteration $i+1$ given that all solutions are genuine at iteration i. A bound on p_{unique} follows by observing that the only genuine solutions of length N are $\mathbf{v}$ and $\mathbf{v}^R$ (Appendix B.2 Theorem 8). Detailed description of our derivation can be found in Appendix B.2.

Our derivation uses the underlying distribution on the database as an abstract parameter. We can then computationally determine concrete values for the bound by instantiating with concrete distributions.

Complexity Analysis. We find worst-case complexity analysis uninformative, as there are cases where the solution set is exponentially large throughout the execution of the algorithm. Furthermore, the worst case scenarios would impact even an average-case analysis. Instead, we concentrate on bounding the probability that the number of partial solutions maintained does not exceed a certain size. In turn, this bound implies a bound on the total runtime of the attack.

It turns out that we can reuse the analysis of uniqueness discussed above. We are interested in the same event as our analysis of uniqueness, that is, we bound the probability that the set of partial solutions of length b to N contain only genuine solutions. As there are at most $2(N-i+1)$ genuine solutions of length i, we know that if the event happens, the number of solutions of length b to N cannot exceed $2N$. On the other hand, the number of solutions before iteration b cannot exceed $|W|^{b-1}$. Therefore, we conclude that the number of partial solutions never exceed $|W|^{b-1}$ with probability p_{unique}.

As a consequence, the space complexity of the attack is $\mathcal{O}(|W|^{b-1})$ with probability p_{unique}. In each iteration of the attack, every partial solution is appended with all volumes in W to the left and to the right, and at most b conditions are checked. Therefore, extending a partial solution takes $\mathcal{O}(2b|W|)$ operations, and the total time complexity is $\max_i \mathcal{O}(bN|\mathcal{S}_i|) = \mathcal{O}(bN|W|^b)$ with probability p_{unique}.

2.3 Experimental Validation

We present an experimental evaluation of our basic attack described in the previous section. We perform attacks on the attributes `AMONTH`, `NDX`, `NPR` and `AGE` as these attributes have a variety of N ranges and different distributions,

allowing us to understand the effectiveness of our attack on different databases. We excluded experimental results on the attribute `LOS`, as most of the databases on the attribute cannot be recovered uniquely with a relatively small b. We discuss how to relax the adversarial goal to recover a part of the database uniquely in Section 5. We also give numerical examples and simulations to the analysis of uniqueness of solutions presented in Section 2.2.

Experimental results. For efficiency of implementation, we abort as soon as the size of the solution set exceeds $|W|^2$. Although our earlier analysis suggests that the solution size can grow exponentially, our experiments show a small threshold is sufficient for most attacks. We test our attack on the attributes and parameters shown in Table 1. For each attribute in turn, we report on the fraction of databases we recover uniquely and the fraction of databases for which more than one solution exist. We also indicate the fraction of databases for which our clique-finding initialization procedure fails and the the fraction of databases where our attack runs out of space.

For the attributes with moderate N, our attack works sufficiently well with bounds as small as 3 to 5. Our attack is less effective on the attribute `AGE` as the databases on the attribute often contain segments of small volumes that cannot be recovered uniquely given the bounds. Nonetheless, even here our attack recovers uniquely (up to reflection) one in five databases.

Attribute	b	Unique	Ambiguous	Clique fail	Abort	Avg. time (s)
AMONTH	3	78.5%	3.2%	0.0%	18.3%	0.0087
NDX-16	3	87.3%	2.1%	0.0%	10.7%	0.0083
NDX-26	4	82.7%	4.1%	0.3%	12.9%	0.0494
NDX-26	5	88.3%	40.0%	1.2%	6.4%	0.0498
NPR-16	3	89.1%	10.2%	0.0%	0.7%	0.0011
NPR-26	4	81.2%	15.2%	0.0%	3.7%	0.0124
NPR-26	5	84.6%	14.2%	0.0%	1.2%	0.0131
AGE	6	22.4%	0.9%	41.1%	35.7%	17.60
AGE	8	32.1%	0.6%	52.6%	14.7%	15.99

Table 1: Experimental data for the basic attack.

Theoretical analysis of uniqueness of solutions and simulation. We perform simulation to study the empirical probabilities that our algorithm returns a unique solution under different parameters, using the theoretical bound described in Section 2.2. We study two types of synthetic distributions for $N = 40$ and b between 5 and 8. Detailed experimental results can be found in Appendix B.3. Our theoretical analysis of the uniqueness of solutions can be overly pessimistic with small b, but it is fairly accurate for larger b. Our experiments provide some initial evidence that databases with large variations of elementary volumes are susceptible to volume leakage attacks.

3 Simple Variations on the Leakage

In this section, we study the robustness of our basic attack with variations on the basic leakage function in Equation (2). These variations correspond to more realistic query distributions, the use of countermeasures, or both. For example, we look at the case where the user may issue some queries with window size larger than b, or decide not to query some of the small ranges. Additionally, the server can pad some fake records to the query responses to invalidate our basic attack. Recall that our basic attack incrementally extends solutions using volumes which need to verify some constraints so additional volumes, or missing ones, will impact our algorithm. Nonetheless, we show that many of the databases can still be reconstructed uniquely under these variations.

3.1 Spurious Volumes

First, we consider two types of spurious volumes in our attack, namely volumes from queries with larger windows and random volumes. We show that with slight modification to our basic attack, a significant proportion of the databases from the HCUP dataset can still be recovered uniquely.

Attack Overview. To model additional volumes we use some distribution $\tilde{Z}$. We write $z \leftarrow \tilde{Z}(\mathbf{v})$ for sampling a volume from this distribution. Notice that we allow the distribution to depend on the real database counts: this is useful to model both the case when noise consists of real volumes (due to real queries outside of unexpected window size) and noise which is calibrated with respect to the real data. The leakage function in the presence of such noise is

$$\mathcal{L}_{b,\tilde{Z},I}(\mathbf{v}) = \mathcal{L}_b(\mathbf{v}) \cup \{z_i \mid i \leq I, z_i \leftarrow \tilde{Z}(\mathbf{v})\}. \tag{3}$$

for some I which itself may depend on $\mathbf{v}$. The goal of the adversary is to reconstruct database $\mathbf{v}$ given a sample from $\mathcal{L}_{b,\tilde{Z}}(\mathbf{v})$.

For this type of scenarios, we argue that the same strategy (with a minimal change) works. That is, we use the same way to incrementally build solutions by checking the same constraints as before. While all solutions can still be found (if the attack does not abort) the presence of noise may lead to additional solutions which include fake volumes. Worse, since the adversary learns $W \leftarrow \mathcal{L}_{b,\tilde{Z}}(\mathbf{v})$ we cannot sift potential solutions by checking that $\mathcal{L}_b(\mathbf{v}) = W$. We experimentally confirm that the success of the adversary does not drastically degrade.

We do need to change the way we initialize our attack: due to noise our initial solution finding algorithm may fail to return a genuine partial solution. For example, if the maximum observed volume is a fake volume, our clique-finding algorithm will certainly no generate proper partial solutions. To overcome this problem, we run the initial solution finding algorithm, iteratively, starting with different volumes and use the union of all initial solutions found this way as the starting point for the iterative part of the attack.

Our attack in the presence of noise is the same as Algorithm 2, except that we omit the final check on the leakage (line 7 of Algorithm 2) and employ a

slightly more elaborate algorithm for identifying starting partial solutions. We refer to the resulting attack as $\mathcal{A}_2$.

Theoretical Analysis. Since we can no longer check for equality of leakages in $\mathcal{A}_2$, we cannot ensure that any solution $\mathbf{v}'$ output by the attack satisfies $\mathcal{L}_b(\mathbf{v}') = W$, though it must be the case that $\mathcal{L}_b(\mathbf{v}') \subseteq W$. The modified correctness property is established by the following theorem. The proof can be found in Appendix C.1.

Theorem 2 (Correctness of the attack in the presence of noise)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* b *be any natural number less or equal to* N*. For any possible sample from the leakage function* $W \leftarrow \mathcal{L}_{b,\tilde{Z}}(\mathbf{v})$*, let* $\mathcal{S}_N$ *be the output of* $\mathcal{A}_2$*, i.e.* $\mathcal{S}_N = \mathcal{A}_2(W, b, N)$*. Then*

1. $\forall \mathbf{v}' \in W^N, \mathcal{L}_b(\mathbf{v}') \subseteq W \Rightarrow \mathbf{v}' \in \mathcal{S}_N$,
2. $\mathbf{v} \in \mathcal{S}_N$.

Experimental Results. We use `NDX-26` and `NPR-26` as the attributes to perform experiments: since our basic attack succeeded with high probability for these attributes, they offer a good starting point to understand the effects of the noise.

For uniformly distributed random noise, the fake volumes are drawn without replacement from a discrete uniform distribution with lower limit as the minimum observed volume, and upper limit as the maximum observed volume. The minimum observed volume is used as the initial solution to the attacks. We compare recovery rate of our attack with $b = 5$ and noise levels 0.5 and 1, where with noise level α, $\alpha \cdot |\mathcal{L}_b(\mathbf{v})|$ fake volumes are added to $\mathcal{L}_b(\mathbf{v})$. We abort an attack as soon as the solution set is larger than $|W|^2$. The results are summarized in Table 2.

Attribute	α	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-26	0.5	62.3%	15.9%	21.9%	0.72
NDX-26	1	58.1%	16.4%	25.5%	1.70
NPR-26	0.5	61.4%	16.7%	22.0%	0.61
NPR-26	1	60.1%	18.1%	21.7%	1.37

Table 2: Experimental data for the attack with uniform random noise.

For volumes from larger windows, we test our attack on $b = 5$ and larger window of size 8. The minimum observed volume is used as the initial solution. We use noise levels $\alpha = 0.5$ and 0.75 in our experiments, where with noise level α, $\alpha \cdot |\mathcal{L}_8(\mathbf{v}) \backslash \mathcal{L}_5(\mathbf{v})|$ volumes from larger windows are added to $\mathcal{L}_5(\mathbf{v})$. We abort an attack as soon as the solution set is larger than $|W|^2$. The results are summarized in Table 3.

Attribute	α	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-26	0.5	64.5%	14.3%	21.2%	0.46
NDX-26	0.75	60.4%	16.1%	23.5%	1.00
NPR-26	0.5	43.6%	20.1%	36.4%	0.45
NPR-26	0.75	41.8%	20.9%	37.3%	0.61

Table 3: Experimental data for the attack with noise from larger windows.

3.2 Missing Queries

In our basic attack, every volume within a given window size is assumed to be observed by the adversary, so all the constraints within the window size b can be checked. In practice, it is possible that some of the queries are not issued by the user as they are uninteresting. It is also possible that the user/server actively blocks some of the queries in an attempt to defend against volume leakage attacks. We demonstrate that it is still possible to reconstruct the database uniquely, even if some of the volumes from the small windows are missing. For simplicity, within each window of size b, we assume the adversary does not have access to k randomly chosen volumes. Furthermore, we assume that the elementary volumes and all two-way sums of the elementary volumes are always part of the leakage. Our assumptions are somewhat arbitrary but our attack demonstrate that there is a lot of redundancy in the leakage function to cope with missing volumes.

Attack Overview. We start with a description of the leakage function one may observe when some queries are suppressed. Let k be a natural number that is less or equal to $b-2$, let $\mathcal{I}$ be an index set of pairs of labels as follows:

1. $\forall i \leq N, (i,i) \in \mathcal{I}$,
2. $\forall i \leq N-1, (i,i+1) \in \mathcal{I}$,
3. $\forall i \leq N-b, |\{(x,y) \in \mathcal{I} \mid i \leq x \leq y \leq i+b-1\}| \geq \frac{b(b+1)}{2} - k$.

Indexes (x,y) that appear in $\mathcal{I}$ are the queries the adversary observes. Notice that condition (3) allows for a certain number of queries to be missing (more specifically k queries for each individual window of size b.

For a fixed $\mathcal{I}$, the adversary learns the leakage function

$$\mathcal{L}_{b,\mathcal{I}}(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i \mid (x,y) \in \mathcal{I}, y-x+1 \leq b \right\}. \tag{4}$$

The adversary is given the leakage of some database $\mathcal{L}_{b,\mathcal{I}}(\mathbf{v})$ and k, and his goal is to reconstruct all $\mathbf{v}$ that generates the set of observed volumes, potentially with a different index set that satisfies all the constraints.

We can no longer check all the additive constraints within the windows, but given our assumption on $\mathcal{I}$, we know that a solution is not plausible if in any window, there are more than k missing constraints. Therefore, we extend a

solution by some new volume only if at most k of the constraints associated to the new volume are missing. Our attack is described by Algorithm 4 in Appendix C, and we call the attack $\mathcal{A}_3$.

Theoretical Analysis. We say that $\mathcal{A}_3$ is correct if for all solutions found by the algorithm, there exists an index set satisfying all the constraints and the resultant leakage is the same as the leakage from the input. The following theorem establishes that $\mathcal{A}_3$ is correct. The proof can be found in Appendix C.3.

Theorem 3 (Correctness of the attack with missing queries)**.** Let $\mathbf{v}$ be a database, $N = |\mathbf{v}|$ and b be any natural number less or equal to N. Let $\mathcal{I}$ be an index set described above. Let $W = \mathcal{L}_{b,\mathcal{I}}(\mathbf{v})$ and $\mathcal{S}_N$ be the output of $\mathcal{A}_3$, i.e. $\mathcal{S}_N = \mathcal{A}_3(W, b, k, N)$. Then

1. $\forall \mathbf{v}' \in W^N, \forall \mathcal{I}' \in ([N] \times [N])^*, \mathcal{L}_{b,\mathcal{I}'}(\mathbf{v}') = W \Rightarrow \mathbf{v}' \in \mathcal{S}_N$,
2. $\mathbf{v} \in \mathcal{S}_N$.

Experimental Results. We test our attack on the attributes `NDX-26` and `NPR-26`, with the missing queries generated uniformly, and the results are shown in Table 4. A significant proportion of the databases can still be reconstructed uniquely in the setting $b = 6$ and $k = 2$, indicating that banning some of the queries is not an efficacious countermeasure.

Attribute	b	k	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-26	5	1	42.8%	23.7%	33.4%	2.20
NDX-26	6	1	59.4%	16.7%	23.9%	21.31
NDX-26	6	2	40.1%	27.4%	32.5%	40.33
NPR-26	5	1	44.6%	20.5%	34.9%	1.18
NPR-26	6	1	56.1%	15.9%	27.9%	5.97
NPR-26	6	2	40.4%	24.0%	35.5%	11.87

Table 4: Experimental data for attack with missing queries.

3.3 Adding Fake Records

To a large extent, our attacks rely on checking equality of volumes, i.e. if v_1 and v_2 are adjacent volumes, then their sum should be in the set of observed volumes. An obvious defense strategy is to pad the responses with fake records. An entire spectrum of instantiations of this idea is possible. We discuss some considerations in padding strategies, and present an attack which bypasses a plausible instantiation of this countermeasure.

The best defense is to pad *all* answers with a large number of fake records. While there is clearly no effective database reconstruction attack, the scheme is highly inefficient. For efficiency one may choose to pad a small fraction of the queries with a small number of fake records. If the parameters of the padding

strategy are chosen inappropriately, the attacker may be able to recover the true observed volume and use our basic attack to reconstruct the underlying database. Interestingly, the padding strategy plays an important role in the security too. For example, if the size of the padding is generated uniformly at random for each individual query then an attacker may be able to learn the true volumes from the perturbed volumes for some databases. On the other hand, if the size of the padding stays is an (unknown) constant r for all queries, then it is easy to see that the optimal guess of any elementary volume always has an uncertainty of r.

Attack Overview. Let $Z = \{Z_{x,y}\}$ be some noise distribution (which can potentially depend on the indices of the queries and the volumes of the query responses). We formally define the leakage function as

$$\mathcal{L}_{b,Z}(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i + z_{x,y} \mid y - x + 1 \leq b, z_{x,y} \leftarrow Z_{x,y} \right\}. \tag{5}$$

Given the leakage of some database $\mathcal{L}_{b,Z}(\mathbf{v})$ and a description of Z, the adversary is asked to find all databases that can generate the set of observed volumes. We study the case where each query is padded with up to r fake entries selected uniformly and independently. That is, i.e. for all x, y, $Z_{x,y} = \text{Uniform}(1, r)$.

Our attack $\mathcal{A}_4$ begins by guessing the ranges for the observed volumes. A guess is of the form (w_1, w_2) where w_1 is the lower bound and w_2 is the upper bound (inclusive) of the guess. A partial solution is a tuple just like that in the basic attack $\mathcal{A}_1$, except that the entries are ranges of the form (w_1, w_2) as described above. The solution extension procedure uses the same idea as the basic attack, but the constraints checked are changed to if the partial solution can generate the padded volumes. We present the pseudocode of the attack in Appendix C.4 Algorithm 5.

We note that our notion of approximate reconstruction is different from that of the GLMP attack [11]. In their notion, the adversary is given the access pattern leakage of *all* queries and his goal is to guess the label of every record within a certain threshold of error. For our attack, the approximation is on the elementary volumes, and this is the best the adversary can do as the volumes are perturbed.

As the goal of the adversary is to approximately recover the database, he can give up some accuracy in his guess to allow for more databases to be uniquely reconstructed. By that, we mean that if there are two solutions $((w_1, w_2), (w_3, w_4))$ and $((w_1, w_5), (w_3, w_4))$ and $w_5 > w_2$, the adversary can merge the guesses as $((w_1, w_5), (w_3, w_4))$ at a loss of accuracy. In our attack, we achieve this trade-off by allowing relaxed guesses on the observed volumes.

Theoretical Analysis. The solutions in the final solution set $\mathcal{S}_N$ are of the form $((w_1, w_2), \ldots, (w_{2N-1}, w_{2N}))$. We say that the attack is correct if given any solution of that form, there is a database $(\mathbf{v}_1, \ldots, \mathbf{v}_N)$ that can generate the given leakage W and the solution contains the database, i.e. $w_{2i-1} \leq \mathbf{v}_i \leq w_{2i}$ for all $i = 1, \ldots, N$. We formalize the correctness of $\mathcal{A}_4$ as follows. The proof

can be found in Appendix C.5.

Theorem 4 (Correctness of attack with padded queries)**.** Let $\mathbf{v}$ be a database, $N = |\mathbf{v}|$ and b be any natural number less or equal to N. Let $Z = \{Z_{x,y}\}$ be distributions of noises with $Z_{x,y} = \text{Uniform}(1, r)$ for some natural number r. Let $W \leftarrow \mathcal{L}_{b,Z}(\mathbf{v})$, R be some estimations of the true volumes and $\mathcal{S}_N$ be the output of $\mathcal{A}_4$, i.e. $\mathcal{S}_N = \mathcal{A}_4(R, W, b, N, r)$. Then

1. $\forall \mathbf{v}' \in \mathbb{N}^N, W \in \text{supp}(\mathcal{L}_{b,Z}(\mathbf{v}')) \Rightarrow \exists s \in \mathcal{S}_N, \forall s_i, s_i[1] \leq \mathbf{v}'[i] \leq s_i[2]$,
2. $\mathbf{v} \in \mathcal{S}_N$.

Experimental Results. We study the effectiveness of our attack with two sets of attacks on the attributes `NDX` and `NPR` with $b = 5$ and $r = 10$. Our choice of r is arguably small but that is due to the fact that the elementary volumes themselves are small. For instance, for the attribute `NPR-26`, over 54% of the elementary volumes are below 100. As before, we abort if the number of partial solutions is over $|W|^2$.

For the first set of attacks, we aim to reconstruct all elementary volumes with the best possible precision. To do that, we compute R as $R = \{(v - r, v - 1) \mid v \in W\}$ and execute our attack (Algorithm 5). The experimental results are shown in Table 5.

Attribute	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-16	4.5%	16.2%	79.3%	0.92
NPR-16	10.0%	13.1%	76.9%	0.64
NDX-26	1.4%	6.0%	92.6%	2.82
NPR-26	6.7%	8.4%	84.9%	1.25

Table 5: Experimental data for the attack with perturbed volumes, the unique solutions are the most information-theoretically precise solutions.

For the second set of attacks, we trade precision for more unique solutions. We compute R just as before, and iteratively merge ranges $(w_1, w_2), (w_3, w_4)$ in R into (w_1, w_4) if $w_2 \geq w_3$. The experimental results are shown in Table 6.

Attribute	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-16	19.1%	10.8%	70.1%	0.84
NPR-16	19.7%	9.8%	70.5%	0.50
NDX-26	11.1%	6.7%	82.1%	2.10
NPR-26	12.1%	9.2%	78.7%	0.74

Table 6: Experimental data for the attack with perturbed volumes, the guesses on the elementary volumes are relaxed.

An overwhelming proportion of the attacks abort as the databases often contain similar elementary volumes. After perturbing, these volumes can often

be swapped without violating the constraints. Overall, adding fake records is a better countermeasure than the other ones we have considered.

4 Attack on Observed Volumes with Bounded Window Size

All attacks in the literature and our attacks described above require the set of elementary volumes to be part of the observed volumes, so one may suspect that these are absolutely necessary. In this section, we show that this is not the case. We construct a successful adversary which only observes volumes for queries of medium window sizes, i.e. there is a lower bound and an upper bound on the window size. More formally, we define the leakage function as:

$$\mathcal{L}_{a,b}(\mathbf{v}) = \left\{ \sum_{i=x}^{y} v_i \mid \ a \leq y - x + 1 \leq b \right\} \tag{6}$$

for some $0 < a < b \leq N$. For simplicity, we assume b and N are multiples of a. The assumptions are not necessary requirements for our attack to work, though they make our attack simpler to present and understand. We do need however that b is somewhat large compared with a, more specifically that $b > k \cdot a$, for some k. We explain below the role played by this restriction.

4.1 Reconstruction Algorithm

Intuition. We provide an overview of the ideas that go into our attack. For concreteness, let's assume that $a = 3, b = 9, N = 12$. The database counts are $\mathbf{v} = (v_1, \ldots, v_{12})$. Here, and throughout this section, we write $\mathbf{v}[i]$ for v_i, and we write $\mathbf{v}[i, j]$ for $\sum_{k=i}^{j} v_k$. Clearly, $\mathbf{v}[i, j] + \mathbf{v}[j + 1, k] = \mathbf{v}[i, k]$.

Our attack proceeds in two stages. In the first stage we recover the sequence $\tilde{\mathbf{v}}_0 = (\mathbf{v}[1, 3], \mathbf{v}[4, 6], \mathbf{v}[7, 9], \mathbf{v}[10, 12])$ (i.e. disjoint queries with window size a which cover the entire $\mathbf{v}$). To piece together $\tilde{\mathbf{v}}_0$ we observe that the leaked information allows us to check constraints on neighboring entries in $\tilde{\mathbf{v}}$. For example, we have that $\mathbf{v}[4, 6] + \mathbf{v}[7, 9] = \mathbf{v}[4, 9]$ and $\mathbf{v}[4, 9]$ occurs in the leakage (it corresponds to a query with window size 6). Similarly, $\mathbf{v}[4, 6] + \mathbf{v}[7, 9] + \mathbf{v}[10, 12] = \mathbf{v}[4, 12]$ also occurs in the leakage (query with window size 9).

Based on the above observations, we construct $\tilde{\mathbf{v}}_0$ by viewing its entry as elementary volumes and applying our basic search strategy. Every entry needs to satisfy between $k - 1$ and $2k - 2$ constraints, where $k = b/a$[1]. Importantly, notice that from $\tilde{\mathbf{v}}_0$ we can also recover all volumes of the form $\mathbf{v}[1, 3i]$.

Next, we determine volumes of the form $\mathbf{v}[1, 3i + 1]$ by reconstructing the sequence $\tilde{\mathbf{v}}_1 = (\mathbf{v}[1, 4], \mathbf{v}[5, 7], \mathbf{v}[8, 12])$. One can think of $\tilde{\mathbf{v}}_1$ as a variant of $\tilde{\mathbf{v}}_0$ shifted by 1. The first query has window size $a + 1$, and all but last of

[1] k is the "window size" for $\tilde{\mathbf{v}}_0$ – the larger the ration of b to a the more constraints we have available

the subsequent ones have window size a. The last query has size $a-1$. Here, again we use the strategy to incrementally build solutions from shorter ones. In addition to the additive constraints which have to be satisfied by neighboring entries (e.g. that $\mathbf{v}[1,4]+\mathbf{v}[5,7]$ occurs in the leakage) we also use two other types of constraints. Taking $w=\mathbf{v}[1,4]$ as an example, notice that it must be the case that $v[1,3] \leq w \leq v[1,6]$. It also must be the case that the values $\mathbf{v}[1,9]-w=\mathbf{v}[5,9]$ and $\mathbf{v}[1,12]-w=\mathbf{v}[5,12]$ occur in the leakage (since these correspond to queries of window size 5 and 8 respectively whereas the maximum size of a query window size is 9). Similar conditions on size and relation to the entries in the leakage hold for the rest of the entries in $\tilde{\mathbf{v}}_1$.

Finally, we also recover $\mathbf{v}[1,3i+2]$ by reconstructing the sequence $\tilde{\mathbf{v}}_2 = (\mathbf{v}[1,5],\mathbf{v}[6,8],\mathbf{v}[9,12])$ using a similar strategy.

At this point, since we have recovered all volumes of the form $\mathbf{v}[1,3i]$, $\mathbf{v}[1,3i+1]$ and $\mathbf{v}[1,3i+2]$ we can recover almost all entries in $\mathbf{v}$ since $\mathbf{v}[t+1]=\mathbf{v}[1,t+1]-\mathbf{v}[1,t]$. In our example, we can recover the values $(\mathbf{v}[4],\mathbf{v}[5],\ldots,\mathbf{v}[9])$.

To complete the attack we recover the elementary volumes in the windows of size 3 at the start and end of $\mathbf{v}$. This can be done by simply extending $(\mathbf{v}[4],\mathbf{v}[5],\ldots,\mathbf{v}[9])$ to the left and right: since all volumes with window size between 3 and 9 are part of the leakage, we can check 6 constraints on $\mathbf{v}[3]$ and $\mathbf{v}[10]$, and so on.

One difficulty which in the above description is not apparent, is that when reconstructing $\tilde{\mathbf{v}}_0,\tilde{\mathbf{v}}_1,\tilde{\mathbf{v}}_2$ we may obtain more than one solution for each. In brief, we overcome this difficulty by relying on constraints that need to hold between the entries in the three different sequences.

Attack overview. Our attack begins by finding all plausible solutions which consists of consecutive volumes corresponding to queries of window size a. That is, we find all valid $\tilde{\mathbf{v}}_0$ from the example above. Call this set $\mathcal{S}_0$. Next, for every solution in $\mathcal{S}_0$, we find sequences of consecutive volumes: the first one belonging to the interval $\mathbf{v}[1,a]$ and subsequent ones being elementary volumes – this step subsumes roughly identifying $\tilde{\mathbf{v}}_1,\tilde{\mathbf{v}}_2,\ldots$ and (most) of the elementary volumes in $\mathbf{v}$. The result is a set $\mathcal{S}'$ of sequences where the first and last entry correspond to compound entries (i.e .queries of window size greater than 1) whereas all other entries are actual elementary volumes. Finally, for each sequence in $\mathcal{S}'$ we recover the elementary volumes on the sides (and remove the compound entries). All solutions are added to a set of tentative solutions $\mathcal{S}$. Finally, we sift through $\mathcal{S}$ and only keep those entries that generate the leakage observed. Full pseudocode of the attack can be found in Appendix D.1 Algorithm 6.

4.2 Correctness

We say our algorithm is correct if it identifies the set of solutions such that every solution in the set generates the same set of observed volumes as the one given at the start of the attack. For convenience, we call our attack on observed volumes with bounded window size $\mathcal{A}_5$. We establish that the algorithm works as expected, under some further assumptions on the parameters a and b. The

proof of the theorem below can be found in Appendix D.2.

Theorem 5 (Correctness of the attack with bounded window sizes)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* a, b *be natural numbers less or equal to* N *with* $b > 2a$*. Let* $\mathcal{S}$ *be the output of the attack, i.e.* $\mathcal{S} = \mathcal{A}_5(\mathcal{L}_{a,b}(\mathbf{v}), a, b, N)$*. Then*

1. $\forall \mathbf{v}' \in \mathcal{L}_{a,b}(\mathbf{v})^N, \mathcal{L}_{a,b}(\mathbf{v}') = \mathcal{L}_{a,b}(\mathbf{v}) \Leftrightarrow \mathbf{v}' \in \mathcal{S}$,
2. $\mathbf{v} \in \mathcal{S}$.

4.3 Experimental Results

We test our attack on the attributes `NDX-26` and `NPR-26` of the HCUP database extensively, as our basic attack recovers the databases on the attributes uniquely with high success rate so attacking the databases with leakage function $\mathcal{L}_{a,b}$ is informative. At the same time, N for the attributes are large enough so that we can study effectiveness of our attack under a variety of choices of a and b. The threshold before aborting is set as $|W|^2$. The experimental results are shown in Table 7.

Attribute	a	b	Unique	Ambiguous	Abort	Avg. Time (s)
NDX-26	3	9	78.1%	1.2%	20.7%	0.49
NDX-26	4	12	80.2%	1.1%	18.7%	1.02
NDX-26	4	16	85.0%	2.5%	12.5%	1.90
NPR-26	3	9	79.8%	0.5%	19.7%	0.30
NPR-26	4	12	84.9%	0.3%	14.8%	0.68
NPR-26	4	16	91.6%	0.4%	8.0%	1.92

Table 7: Experimental results for the attack on bounded window size.

Our attack on the attribute `NDX-26` with $a = 3$ and $b = 9$ has recovered over 78% of the databases uniquely. This suggests that banning queries from small windows alone is ineffective as a countermeasure. Furthermore, our attack on the attribute `NPR-26` with $a = 4$ and $b = 16$ is able to reconstruct over 91% of the databases uniquely, indicating that the use of larger b makes the databases more vulnerable to volume leakage attacks.

5 Partial Reconstruction

Not all databases are uniquely reconstructable as shown by our previous attacks. However, the information in the observed volumes often allows unique reconstruction of a segment of the database. Consider database $\mathbf{v} = (100, 2, 1, 1, 1, 2, 1)$. The database is not uniquely reconstructable from $\mathcal{L}_3(\mathbf{v})$. However, by setting 100 as the initial solution and run our basic attack, we find $(100, 2, 1)$ (and its reflection) as the only solution of length 3. This means we have uniquely reconstructed $(100, 2, 1)$ as a segment of the database.

We introduce the partial reconstruction problem as follows. The adversary obtains leakage $\mathcal{L}(\mathbf{v})$ for some database $\mathbf{v}$, and his goal is to output a segment of the database $s \in \mathbf{v}$ (up to reflection). There is more than one choice on how to measure the effectiveness of an adversary with respect to this kind of attacks. Of the several different choices on how to quantify the success of the adversary we consider two. The first simply counts the fraction of labels for which the attack recovers the correct counts; the second, also accounts for the size of the counts themselves. For example, an adversary may correctly identify counts for a small fraction of the labels, but the total number of records associated to these labels can be overwhelming.

5.1 Partial Reconstruction Algorithm

Intuition. Our partial reconstruction attack can be viewed as a special case of our basic attack where instead of reporting all solutions of length N, we return the longest solution that can be uniquely identified. Databases that can only be reconstructed partially usually have a segment of small volumes. For example, the length of stay of the patients in hospitals is usually less than 10 days, with occasional longer stays. If we use the minimum observed volume as the initial solution, it is very likely for the number of solutions to grow out of control. Hence, we initialise our partial reconstruction attack with the clique-finding algorithm described by Algorithm 1. Unlike our basic attack, solutions for the partial reconstruction attack have to be extended in one direction at a time. This is because extending the solutions in both directions inherently introduces ambiguity (and we aim to identify a unique common subsequence).

Attack Overview. Our partial reconstruction attack is described by Algorithm 3. We begin by running Algorithm 1 as the initial solution finding algorithm. Reflections are removed from the initial solutions. The set of partial solutions is then extended iteratively by procedures Extend Left and Extend Right of the basic attack. Only the longest partial solution that is unique is kept. For practical purposes, the set of partial solutions is extended as much as possible until some threshold on its size has reached, and the longest unique solution amongst those is kept by the attacker. If a unique solution is found with line 15 of the algorithm, we return the solution in line 16.

5.2 Correctness and Complexity Analysis

Correctness. Correctness of the partial reconstruction attack needs to be formalized differently from other attacks as we do not recover the whole database most of the time. However, whenever we recover a unique solution, it must be a segment of the original database (up to reflection). We call our partial reconstruction attack $\mathcal{A}_6$. The following Theorem establishes the correctness of partial reconstruction attack. The proof is similar to that of the basic attack with the additional constraint that the partial solution is unique so it is omitted.

Algorithm 3 Partial reconstruction attack

input $W = \{\sum_{i=x}^{y} v_i \mid y - x + 1 \leq b\}, b, N$
output $(w_1, \ldots, w_m)$ with $m \leq N$ and for all i, $w_i \in W$

procedure ATTACK(W, b, N)
  $\mathcal{S}_b \leftarrow$ INITIAL SOLUTION(W, b)
  for $s \in \mathcal{S}_b$ **do**
    if $s^R \in \mathcal{S}_b$ **then**
      $\mathcal{S}_b \leftarrow \mathcal{S}_b \backslash \{s\}$
  for $i \leftarrow b + 1, N$ **do**
    $\mathcal{S}_i \leftarrow$ EXTENDLEFT($\mathcal{S}_{i-1}, W, b$)
  $j \leftarrow \max_i\{i \mid |\mathcal{S}_i| = 1\}$
  for $i \leftarrow j + 1, N$ **do**
    $\mathcal{S}_i \leftarrow$ EXTENDRIGHT($\mathcal{S}_{i-1}, W, b$)
  if $\min\{|\mathcal{S}_i|\} > 1$ **then**
    return ()
  $j \leftarrow \max_i\{i \mid |\mathcal{S}_i| = 1, i \geq j\}$
  return $\mathcal{S}_j.pop()$

Theorem 6 (Correctness of the partial reconstruction attack)**.** *Let* $\mathbf{v}$ *be a database,* $N = |V|$ *and* b *be any natural number less or equal to* N*. Let* $s = \mathcal{A}_6(\mathcal{L}_b(\mathbf{v}), b, N)$*. Then* $s \in \mathbf{v}$ *or* $s^R \in \mathbf{v}$*.*

COMPLEXITY. Algorithm 1 takes at most $\mathcal{O}(|W|^b)$ time and space following a standard argument for brute-force clique finding. The solution extension procedures takes at most $\mathcal{O}(|W|^{b-1})$ space and $\mathcal{O}(b{\cdot}N{\cdot}|W|^b)$ time as the clique finding step generates at most $\mathcal{O}(|W|^{b-1})$ solutions. Therefore, the overall space complexity of the attack is $\mathcal{O}(|W|^b)$ and the overall time complexity of the attack is $\mathcal{O}(b \cdot N \cdot |W|^b)$.

In practice, we stop the attack as soon as there are more than $|W|^2$ solutions (including the clique finding step). So the space complexity in practice is $\mathcal{O}(|W|^2)$ and the time complexity is $\mathcal{O}(b \cdot N \cdot |W|^3)$.

5.3 Experimental Results

We choose `LOS` as the attribute to attack from the HCUP dataset since our basic attack performs poorly due to presence of small elementary volumes. In addition to the performance parameters introduced at the start of the section, we measure the fraction of databases for which our attack fails due to the computational threshold of $|W|^2$ or inability to identify a unique solution.

Our experimental results are summarised in Table 8 where we report both the fraction of labels correctly identified (column "Length") and the fraction of correctly identified labels but weighted by their value (column "Volume").

Attribute	b	Length	Volume	Failed	Time (s)
LOS	4	37.5%	95.6%	14.7%	0.31
LOS	6	48.1%	97.3%	14.2%	0.99
LOS	8	54.5%	98.2%	15.3%	2.24
LOS	10	59.0%	98.7%	16.5%	4.54

Table 8: Experimental results for the partial reconstruction attack.

Our partial reconstruction attack works on over 85% of the databases on `LOS` for bound b as small as 4. Although only 37.5% of the labels can be recovered uniquely on average, it corresponds to 95.6% of the inpatients. This means that the partial reconstruction attack has effectively recovered the majority of the entries of the databases uniquely. The attack is very efficient despite the expensive clique-finding procedure.

6 Use of Side Information

Side information on the underlying databases can be used to boost our attacks. In this section, we discuss and experiment with two general ways of using side information

Post-processing. The most straightforward use of side information is to post-process the results of our attacks to further sift the possible solutions output by our attacks. We note that the solutions output by our attacks are naturally ambiguous as it is information-theoretically impossible to distinguish the database from its reflection. With the help of minimal knowledge on the database, the attacker may be able to distinguish the database from its reflection. For instance, over 92% of the databases in HCUP, for the attribute `AGE` it holds that that $\mathbf{v}[2,5] < \mathbf{v}[N-4, N-1]$ so almost all databases can be distinguished from their reflections. More interestingly, if the adversary has access to more concrete side information, he may be able to identify the real database from a large set of solutions. For example, if the attacker knows some true volumes of some segments of the database, he can use that information to disambiguate the solutions.

Dynamic. Side information can also be used dynamically to prune the search space more effectively and reduce the number of ambiguous solutions. For example, over 50% of the databases on the attribute `NPR-26` satisfies the condition $\mathbf{v}[i] > \mathbf{v}[i+1]$ for $2 \leq i \leq N-1$. This means if the attacker knows the target database has this property, the partial solutions must be decreasing in the middle, and the volumes used to extend the partial solutions to the left must be larger than the first volume of the partial solution and vice versa.

We experimentally verify the effect of side information on two of our attacks. We attack the attribute `NDX-26` in the setting of missing queries, with the same parameters as those in Table 4. In addition to the observed volumes and relevant parameters to the attack, the adversary is given the volume of the answer to

up to 3 queries of window size within b. The queries are generated uniformly at random. For each hospital, we run the attack 10 times with freshly selected known queries, and report the average rate of success. We observe a considerable increment in the uniqueness rate of solutions as compared to the case where the attacker has received no additional information (Table 4), as shown by Table 9.

		Unique			
b	k	$m = 1$	$m = 2$	$m = 3$	Abort
5	1	51.8%	55.7%	57.7%	34.0%
6	1	64.7%	67.5%	68.9%	24.9%
6	2	50.2%	53.6%	55.9%	34.3%

Table 9: Experimental data for attack with missing queries on the attribute `NDX-26`. The attacker is given $m = 1, 2, 3$ random known queries.

On the same attribute, we have also tested known $\mathbf{v}[1]$ and $\mathbf{v}[N]$ as the side information, and the results are shown in Table 10. The uniqueness rates of solutions are higher than those in the previous setting. This hints that the ambiguity in the solutions often comes from the initial and final segments of the databases.

b	k	Unique	Ambiguous	Abort
5	1	63.3%	2.4%	34.3%
6	1	72.9%	1.8%	25.3%
6	2	64.7%	1.6%	33.7%

Table 10: Experimental data for attack with missing queries on the attribute `NDX-26`. The attacker is given $\mathbf{v}[1]$ and $\mathbf{v}[N]$.

We attack the attribute `NPR-26` in the setting of perturbed volumes, with $b = 5$ and $r = 10$. The attack is conducted on the databases with the property $\mathbf{v}[i] > \mathbf{v}[i+1]$ for $2 \leq i \leq N-1$, and this information is given to the attacker. The side information improves the fraction of the databases that can be reconstructed uniquely from 12.1% (Table 6) to 24.8%.

7 Countermeasures

Our attacks show that left unchecked volume leakage can be devastating. Here we discuss two generic types of countermeasures which have been proposed from the point of view of our attacks.

Modifying the database. One modification to consider is to group labels into buckets [14]. This is commonly known as bucketization technique. If, as natural, this is done with consecutive values, then volume leakage attacks can now target bucketed elementary volumes. So such technique only prevents the adversary from learning the original elementary volumes, but our attack still apply and

can recover the bucket sizes. Alternatively, one can add dummy records. Intuitively, a small number of dummy records does not change the database much so either padding with dummy records leads to (space/bandwidth) inefficient solutions, or as shown by our attack which use noisy leakage, significant amount of information about the database may still be recovered. We also note that using a differential private mechanism to add dummy records as suggested by KKNO [16] is not effective in our setting. Differential privacy only protects the database from exact reconstruction, but the "shape" of the database is not changed by the mechanism.

Modifying query processing. Queries can be processed to hide certain volumes. Two prior suggestions due to Grubbs et al. [12] are to batch queries together and therefore hide all elementary volumes. In particular, one can restrict the minimum window size of a range query. Our attack in Section 4 suggests that these are not effective countermeasures.

New proposals. Informed by our attack (or rather by the cases where our attacks fail) we suggest two directions for research on countermeasures against volume leakage attacks. Here, we consider the less ambitious but still challenging setting where the server is honest and we are only trying to protect the scheme from passive external observers. The first proposal is to pad the answers it returns so that all volumes observed are powers of two. Suppose $\{2^i, 2^j, 2^k, 2^l\} \in W$, with $i < j < k < l$ and $l = k + 1$. Then the following tuples (up to reflection) can all be part of the original database: $(2^i, 2^k), (2^j, 2^k), (2^k, 2^k)$. The intuition is that the leakage does not contain any information regarding the adjacency of the indexes queried and our attack strategy cannot rely on any obvious constraints. In the worst case however, the bandwidth required by the scheme doubles.

A preliminary idea for a method which incurs less bandwidth loss is based on the following observation. Imagine that one can somehow ensure that the leakage function is $\mathcal{L}(\mathbf{v}) = \left\{\sum_{x \in X} x \mid X \subseteq \mathbf{v}\right\}$, where by $X \subseteq \mathbf{v}$ we mean that X is a subset of the set of entries in $\mathbf{v}$. That is, the leaked volumes comprise all the sums of the subsets of the elementary volumes in the database. It is immediate in this case that all possible permutations of $\mathbf{v}$ are preimages of $\mathcal{L}(\mathbf{v})$ so short of degenerate cases (and of using side information about the database) it is impossible to uniquely recover $\mathbf{v}$. The technical reason why our recovery algorithm fails in this case is that leakage function contains spurious volumes which have "the right" structure: when extending partial solutions we check if certain sums of volumes occur in W but his check will always succeed. A natural countermeasure inspired by the above discussion is to ensure that the information leaked contains as many possible volumes from $\mathcal{L}(\mathbf{v})$ above: these increase the likelihood that the check we made when extending solutions succeeds even for invalid extensions, and therefore has the potential to disrupt our attacks.

We emphasize that this is a rather preliminary idea: we cannot ensure that the leakage is the entire $\mathcal{L}(\mathbf{v})$ above (there are exponentially many entries in $\mathcal{L}(\mathbf{v})$ so how to pad appropriately is still a problem which we leave for future

research.

8 Conclusion

We present new volume leakage attacks against encrypted range queries. We improve over the state of the art in several respects. Our attacks require significantly less data, are robust when the leakage includes noise, and can be extended to deal with other practically relevant settings. We show how to deal with less orthodox leakage functions (e.g. only medium window size queries) and identify formalize partial reconstruction attacks, a realistic adversarial goal. We discussed how side information can be used to improve the effectiveness our attacks further. Our highly effective attacks spotlight the rich structure of W as a key source of insecurity and explain the difficulty of coming up with efficient countermeasures. We suggest some preliminary ideas in this direction, but leave a careful study and deployment for future work.

The key idea underlying our attacks can also be extended to deal with non-dense databases. For the leakage in our basic attack, zeros in the databases essentially generates less constraints when the solutions are extended, and this information can be used to locate the zeros. We illustrate this with an example. Consider $\mathbf{v} = (4, 2, 5, 0, 7)$ and $b = 4$, if we try to extend the partial solution $(4, 2, 5)$ to the right with 7, we are only able to identify 3 additive constraints as the two-way sum cannot be checked. This tells us that the partial solution can be extended with $(0, 7)$.

Finally, we highlight a difficulty faced by both design and cryptanalytic work in the space of searchable encryption. Essentially all works in this space either make assumptions on the query distribution (which are often difficult to justify) or study worst case scenarios. What is missing is a principled description/understanding of the query distributions encountered in practice: this would both help to further validate cryptanalysis work and open up the space for designs of schemes and countermeasures which can take advantage of this knowledge.

References

[1] R. Arratia, L. Goldstein, and L. Gordon. Two moments suffice for poisson approximations: The chen-stein method. *Ann. Probab.*, 17(1):9–25, 01 1989.

[2] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 217–232. Springer, Heidelberg, August / September 2006.

[3] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

[4] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Catalin Rosu, and Michael Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *NDSS 2014*. The Internet Society, February 2014.

[5] Louis H. Y. Chen. An approximation theorem for sums of certain randomly selected indicators. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 33(1):69–74, Mar 1975.

[6] Louis H. Y. Chen. Poisson approximation for dependent trials. *Ann. Probab.*, 3(3):534–545, 06 1975.

[7] Carlo Curino, Evan Philip Charles Jones, Raluca Popa, Nirmesh Malviya, Eugene Wu, Samuel Madden, Hari Balakrishnan, and Nickolai Zeldovich. Relational cloud: A database-as-a-service for the cloud. pages 235–240, 04 2011.

[8] Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 79–88. ACM Press, October / November 2006.

[9] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. A secure channel free public key encryption with keyword search scheme without random oracle. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 248–258. Springer, Heidelberg, December 2009.

[10] Agency for Healthcare Research and MD. Quality, Rockville. Hcup nationwide inpatient sample (nis). healthcare cost and utilization project (hcup). `https://www.hcup-us.ahrq.gov/nisoverview.jsp`, 2018. Accessed: 2019-02-20.

[11] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson. Learning to reconstruct: Statistical learning theory and encrypted database attacks. In *2019 2019 IEEE Symposium on Security and Privacy (SP)*, volume 00, pages 480–496, 2019.

[12] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 315–331. ACM Press, October 2018.

[13] H. Hacigumus, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proceedings 18th International Conference on Data Engineering*, pages 29–38, Feb 2002.

[14] Bijit Hore, Sharad Mehrotra, and Gene Tsudik. A privacy-preserving index for range queries. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 720–731. VLDB Endowment, 2004.

[15] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1329–1340. ACM Press, October 2016.

[16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Accessing data while preserving privacy, 2017.

[17] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson. Improved reconstruction attacks on encrypted data using range query leakage. In *2018 IEEE Symposium on Security and Privacy*, pages 297–314. IEEE Computer Society Press, May 2018.

[18] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.

[19] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In *NDSS 2014*. The Internet Society, February 2014.

[20] Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*, pages 583–602, Berkeley, Calif., 1972. University of California Press.

A Experimental Data and the Attacks

Dataset. The Agency for Healthcare Research and Quality (AHRQ) is the lead Federal agency in US with a mission to improve the safety and quality of America's healthcare system. The Healthcare Cost and Utilization Project (HCUP) is a family of healthcare databases and related software tools and products developed and sponsored by AHRQ. One of the datasets in the HCUP is the National (nationwide) Inpatient Sample (NIS), which is the largest publicly available all-payer inpatient healthcare database in US. It contains data from

over 7 million hospital stays each year. We use the HCUP dataset to mean the NIS throughout our paper.

The NIS has been anonymised to protect patient privacy. **We did not attempt to deanonymise any of the data, nor are our attacks designed to deanonymise medical data.** All authors have completed the HCUP Data Use Agreement training and submitted signed Data Use Agreements to the HCUP Central Distributor.

We use the HCUP dataset from year 2004, 2008 and 2009 for our experiments. For each year the dataset contains data for about one thousand hospitals. The median number of inpatients is around 3 to 4 thousand depending on the year. Most of the data fields are for discharge information and hence, not suitable for our attacks. We choose admission month (`AMONTH`), number of diagnoses (`NDX`), number of procedures (`NPR`), age (`AGE`), and length of stay (`LOS`) as the target attributes for our attacks as these attributes are suitable for range queries and they have diversified distributions. For the attributes `NDX` and `NPR`, there are 16 categories for year 2004 and 2008 and these attributes are labelled as `NDX-16` and `NPR-16` respectively. For the same attributes, there are 26 categories for year 2009 and these attributes are labeled as `NDX-26` and `NPR-26` respectively. Summary statistics of the attributes can be found in Table 11. We also offer a comparison of the number of queries required to observe all volumes in the leakage for the GLMP attack and our basic attack with a typical window size, assuming an uniform query distribution.

Attribute	Max. N	Avg. N	Dense	b	GLMP	Basic
AMONTH	12	11.9	97.8%	3	451.2	149.1
NDX-16	16	12.0	4.3%	3	598.4	175.0
NDX-26	26	17.2	8.0%	4	1221.3	341.6
NPR-16	16	9.2	15.3%	3	343.2	113.2
NPR-26	26	10.7	1.0%	4	463.8	168.7
AGE	83	71.1	39.2%	6	16666.3	1714.5
LOS	366	45.9	0.0%	6	6671.6	805.9

Table 11: Summary statistics on the attributes used in the attacks.

Data Processing. For each year, attribute and hospital, we build a database by merging the records together, and remove the labels with zero counts. For the attribute `AGE`, we top-code at 90 in all experiments just like the GLMP attack. Furthermore, we merge ages 1 to 5 and 6 to 10 into two labels for our basic attack as many databases cannot be uniquely reconstructed with small bound on the window size due to this segment of the database. The remaining databases has $N = 83$ and we believe that it is sufficiently large to demonstrate efficiency and effectiveness of our attacks.

We define an attack as the process of generating some set of observed volumes specified by some leakage function for a database on an attribute and a hospital, and running one of our algorithms. Some of the leakage functions we have

considered are probabilistic in nature. In that case, we run the attacks with randomized realizations of the leakage functions. To test the performance of our attack on a leakage function, we run the attack on individual attributes for each hospital at a time and report performance metrics regarding the success probability.

Attack Environment. We implement data processing and the attacks in python and theoretical analysis of our basic attack in R. The attacks are run on an Intel Core i7 4.7GHz machine with 16GB of DDR4 RAM clocked at 2133MHz.

Performance Parameters of the Attacks. There are five performance parameters that are common to most of our attacks, namely the fraction of databases which we can recover uniquely (up to reflection), the fraction of databases for which we obtain ambiguous solutions, the fraction of databases where the initial solution finding algorithm fails, the fraction of databases for which the attack aborts, and the average time taken for the attack on the given attribute across all databases.

B Basic Attack

B.1 Correctness of Basic Attack

Proof of Theorem 1. Statement (2) follows from (1), so it suffices to prove (1).

($\Leftarrow$) We obtain this implication for free as line 7 of the algorithm ensures that $\mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v})$.

($\Rightarrow$) Assume that $\mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v})$. We show that at iteration i of the algorithm in line 6, we have the invariant that $\forall \mathbf{v}' \in \mathcal{L}_b(\mathbf{v})^N$, $\mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v}) \implies \exists (x_1, \ldots, x_i) \in \mathbf{v}'$ s.t. $(x_1, \ldots, x_i) \in \mathcal{S}_i$. In particular, this implies that for solutions of length N, $\mathbf{v}' \in \mathcal{S}_N$. We prove this by induction.

The correctness of the base case relies on the correctness of the initial solution finding algorithm. We prove that Algorithm 1 is correct. Assume for the sake of contradiction that there exists some clique of size b, $(w_1, \ldots, w_b)$ which is not in the output of the algorithm. Then $\left\{w_1, w_1 + w_2, \ldots, \sum_{j=1}^{b} w_j\right\}$ must not be in G_b. This is impossible as $\left\{w_1, \sum_{j=1}^{b} w_j\right\} \in G_2$ and for i between 3 and b, adding $\sum_{j=1}^{i} w_j$ to the previous set of volumes generates a valid element in G_i. Therefore, our assumption must be false and the correctness of the clique finding algorithm follows.

In the inductive step, we assume that at iteration k, there exists $(x_1, \ldots, x_k) \in \mathbf{v}'$ such that $(x_1, \ldots, x_k) \in \mathcal{S}_k$. Without loss of generality, we can assume x_k is not the last volume to the right of $\mathbf{v}'$, so there is some x_{k+1} such that $(x_1, \ldots, x_k, x_{k+1}) \in \mathbf{v}'$. Running the Extend Right procedure from line 15 to 20, we check if $\left\{x_{k+1} + \sum_{i=0}^{j} x_{k-i} \mid \ 0 \leq j \leq \min\{b-1, k\}\right\} \subseteq \mathcal{L}_b(\mathbf{v})$. This is indeed the case as $\mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v})$. So there exists $(x_1, \ldots, x_k, x_{k+1}) \in \mathbf{v}'$ such that $(x_1, \ldots, x_k, x_{k+1}) \in \mathcal{S}_{i+1}$.

Hence, we conclude that after the final iteration, $\forall \mathbf{v}' \in \mathcal{L}_b(\mathbf{v})^N$, $\mathcal{L}_b(\mathbf{v}') = \mathcal{L}_b(\mathbf{v}) \implies \exists (x_1, \ldots, x_N) \in \mathbf{v}'$ s.t. $(x_1, \ldots, x_N) \in \mathcal{S}_i$, and the desired result follows. □

B.2 Uniqueness of Solution of Basic Attack

The probability that the final solution is unique is computationally infeasible to compute from its analytical expression so we use a series of bounding and approximation techniques to find an estimation of the value. Given a database $\mathbf{v}$, we call a solution s *genuine* if $s \in \mathbf{v}$ or $s^R \in \mathbf{v}$. We make the following observation. If we discover a solution that is not genuine at any stage of the algorithm, then there must exist a tuple of observed volumes that is not a segment of $\mathbf{v}$ but still satisfies all the constraints. If we can estimate the probability of a random tuple satisfying all the constraints, then we can derive the probability that there is no combination of the observed volumes other than the genuine solutions by the end of the attack.

Our estimation of p_{unique} is derived using the following lemmas. In Lemma 1, we derive the distribution of observed volumes. This allows us to work out the distribution of out-of-order sums of the observed volumes in Lemma 2. Combining the two lemmas, we find an estimation on the probability that the solutions of length b contains only genuine solutions using Lemma 3. Finally, we derive an estimation of p_{unique} using Theorem 8.

Distribution of observed volumes. We write $\tilde{W}$ to denote the distribution of observed volumes. Unlike a simple random variable, a draw from $\tilde{W}$ generates a set of volumes. Given the distribution of elementary volumes as $\tilde{V} = (V_1, \ldots, V_N)$, and bound b, a draw from $\tilde{W}$ is equivalent to $\{\sum_i^j v_i \mid \ v_i \leftarrow V_i, j - i + 1 \leq b\}$. Volumes in $\tilde{W}$ have significant dependencies amongst each other, and it is not trivial to compute $\tilde{W}$. We appeal to the works of Stein [20] and Chen [6, 5], later known as the Chen-Stein method. It works as follows.

Let I be a finite or countable index set. For each $\alpha \in I$, let X_α be a Bernoulli random variable with $p_\alpha = \Pr[X_\alpha = 1] > 0$. Let

$$U = \sum_{\alpha \in I} X_\alpha \text{ and } \lambda = \mathbb{E}(U). \tag{7}$$

We assume $\lambda \in (0, \infty)$. Let Z be a Poisson random variable with the same mean as U. For each $\alpha \in I$, we define $B_\alpha \subset I$ with $\alpha \in B_\alpha$ as the neighbourhood of X_α.The neighbourhood of X_α is the set of random variables that have dependency with X_α. Define

$$b_1 = \sum_{\alpha \in I} \sum_{\beta \in B_\alpha} p_\alpha p_\beta, \tag{8}$$

$$b_2 = \sum_{\alpha \in I} \sum_{\alpha \neq \beta \in B_\alpha} p_{\alpha\beta}, \text{ where } p_{\alpha\beta} = \mathbb{E}(X_\alpha X_\beta) \text{ and} \tag{9}$$

$$b_3 = \sum_{\alpha \in I} \mathbb{E}(\mathbb{E}(X_\alpha - p_\alpha \mid \ \sigma(X_\beta : \beta \neq B_\alpha))). \tag{10}$$

Loosely, b_1 measures the neighbourhood size, b_2 measures the expected number of neighbours of a given occurrence and b_3 measures the dependence between an event and the number of occurrences outside its neighbourhood.

The main result we use is proven by Arratia, Goldstein and Gordon [1].

Theorem 7. *Let $U = \sum_{\alpha \in I} X_\alpha$ be the number of occurrences of dependent events, and let Z be a Poisson random variable with $\mathbb{E}(Z) = \mathbb{E}(U) = \lambda < \infty$. Then*

$$\begin{aligned}
&\sup_x |\Pr[U = x] - \Pr[Z = x]| \\
\leq &2\left[(b_1 + b_2)\frac{1 - e^{-\lambda}}{\lambda} + b_3 \min(1, 1.4\lambda^{-1/2})\right] \\
\leq &2(b_1 + b_2 + b_3),
\end{aligned}$$

and,

$$\begin{aligned}
&\left|\Pr[U = 0] - e^{-\lambda}\right| \\
\leq &(b_1 + b_2 + b_3)(1 - e^{-\lambda})/\lambda \\
< &\min(1, \lambda^{-1})(b_1 + b_2 + b_3).
\end{aligned}$$

For our purpose, X_α's are indicator random variables that tell us if the sum of adjacent elementary volumes associated with the index α takes a particular value u or not. These events usually happen with small probability so it is reasonable to use Poisson approximation.

Let $\tilde{V} = (V_1, \ldots, V_N)$ be the distribution of elementary volumes and b be the maximum window size. We define the index set as $I = \{(i, j) \mid \ j - i + 1 \leq b\}$. We define the neighborhood set of index (i, j) as $\beta_{(i,j)} = \{(k, l) \mid i < k < j, l > j, l - k + 1 \leq b\} \cup \{(k, l) \mid k < i, i < l < j, l - k + 1 \leq b\}$. We define $X_{(i,j)}(u) = \mathbb{1}\left\{\sum_{k=i}^{j} V_k = u\right\}$ to be the indicator random variables to check if the sum of i-th elementary volume through j-th elementary volume is equal to some volume u. Then $U(u) = \sum_{\alpha \in I} X_\alpha(u)$ is the random variable for the number of times we see u in the set of observed volumes. Using second part of Theorem 7, we find a lower bound on the probability that we do not see u in the set of observed volumes, and hence an upper bound on the probability that we do see u. The result is summarized in the following lemma.

Lemma 1 (Distribution of Observed Volumes)**.** *Let $\tilde{V} = (V_1, \ldots, V_N)$ be the distribution of elementary volumes and b be the maximum window size. Let I be the index set, $\beta_{(i,j)}$ be the neighborhood set, and $X_{(i,j)}(u)$ be the indicator random variables defined above. Let $U(u) = \sum_{\alpha \in I} X_\alpha(u)$. Then*

$$\Pr[U(u) > 0] \leq 1 - e^{-\lambda(u)} + \min(1, \lambda^{-1})(b_1 + b_2),$$

where $\lambda(u), b_1$ and b_2 are quantities defined in Equation (7)*,* (8) *and* (9) *respectively.*

Proof. Lemma 1 is a direct result from Theorem 7. For our purpose, $b_3 = 0$, so it suffices to find $\lambda(u), b_1$ and b_2.

It is straightforward to compute $\lambda(u)$ for volume u from definition. We have

$$\begin{aligned}\lambda(u) =& \mathbb{E}(U(u)) && (11)\\ =& \mathbb{E}\left(\sum_{\substack{(i,j)\\ j-i+1\le b}} X_{(i,j)}(u)\right) && (12)\\ =& \sum_{\substack{(i,j)\\ j-i+1\le b}} \Pr\left(\sum_{k=i}^{j} V_k = u\right). && (13)\end{aligned}$$

Equation (11) is a rephrasing of $\lambda(u)$. Equation (12) expands equation (11) by writing out the indicator random variables explicitly. Equation (13) uses linearity of expectation and rewrites the indicators as probabilities.

For index (i, j), the neighbourhood set is every pair of (k, l) such that (i, j) and (k, l) overlaps. Though we know that if $\sum_{m=i}^{j} V_m = u$, then there is no way $\sum_{m=k}^{l} V_m = u$, if $k < i$ and $l > j$, and vice versa. So k or l has to be between i and j, except when $k = i, l = j$. Therefore we can derive $b_1(u)$ for volume u as follows.

$$\begin{aligned}b_1(u) =& \sum_{\alpha\in I}\sum_{\beta\in B_\alpha} p_\alpha(u)p_\beta(u)\\ =& \sum_{\substack{(i,j)\\ j-i+1\le b}} p_{(i,j)}(u)^2 + \sum_{\substack{(i,j)\\ j-i+1\le b}}\sum_{\substack{(k,l)\\ i<k<j, l>j\\ l-k+1\le b}} p_{(i,j)}(u)p_{(k,l)}(u)\\ &+ \sum_{\substack{(i,j)\\ j-i+1\le b}}\sum_{\substack{(k,l)\\ k<i, i<l<j\\ l-k+1\le b}} p_{(i,j)}(u)p_{(k,l)}(u),\end{aligned}$$

where $p_{(i,j)}(u) = \Pr\left[\sum_{k=i}^{j} V_k = u\right]$.

To derive $b_2(u)$ for volume u, it suffices to derive an expression for $p_{\alpha\beta}(u)$. Without loss of generality, we can assume $\alpha = (i, j)$, $\beta = (k, l)$, and $i < k < l$.

Then

$$
\begin{aligned}
& p_{(i,j)(k,l)}(u) \\
&= \Pr\left(X_{(i,j)}(u) X_{(k,l)}(u)\right) \\
&= \Pr\left(\sum_{m=i}^{j} V_m = u, \sum_{m=k}^{l} V_m = u\right) \\
&= \sum_{x} \Pr\left(\sum_{m=i}^{k-1} V_m = x\right) \cdot \Pr\left(\sum_{m=k}^{j} V_m = u - x\right) \cdot \Pr\left(\sum_{m=j+1}^{l} V_m = x\right).
\end{aligned}
$$

So

$$
\begin{aligned}
b_2(u) = \sum_{\substack{(i,j) \\ j-i+1 \le b}} \sum_{\substack{(k,l) \in \beta_{(i,j)} \\ (k,l) \neq (i,j)}} \sum_{x} & \Pr\left(\sum_{m=i}^{k-1} V_m = x\right) \cdot \\
& \Pr\left(\sum_{m=k}^{j} V_m = u - x\right) \cdot \Pr\left[\sum_{m=j+1}^{l} V_m = x\right].
\end{aligned}
$$

Finally, by combining the quantities above using Theorem 7, we obtain the desired result. □

Distribution of out-of-order sum of observed volumes. An out-of-order sum of the observed volumes is a summation of volumes picked from the set of observed volumes in random order. So the sum does not necessarily correspond to a sum of adjacent elementary volumes. We need the distribution of out-of-order sum of the observed volumes because it is key to derive the probability that a random sum of the observed volumes is equal to one of the observed volumes.

To derive the distribution, we do not find the Chen-Stein method as useful as the neighbourhood set is always everything, and b_1 and b_2 are too large to provide any meaningful bound. Instead, we simply use union bound for this purpose. Below, we write U_i for the distribution of the summation of i volumes from the set of observed volumes, where the database is distributed with $\tilde{V} = (V_1, \ldots, V_N)$. By abusing notation, we write U to mean the distribution of the observed volumes.

Lemma 2 (Distribution of out-of-order sum of observed volumes)**.** *Let $\tilde{V} = (V_1, \ldots, V_N)$ be the distribution of elementary volumes and b be the maximum window size. Write U for the distribution of observed volumes and U_k for the distribution of summations of k volumes from the set of observed volumes. We have*

$$\Pr[u \in U_k] \approx \Pr[U * \ldots * U = u] \ / \ i!,$$

where $U * \ldots * U$ is the convolution of U k times.

Proof.

$$\Pr[u \in U_i] \tag{14}$$

$$= \Pr\left[\bigcup_{u_1,\ldots,u_{i-1}} u_1 \in U, \ldots, u_{i-1} \in U, u - \sum_{j=1}^{i-1} u_j \in U\right] \tag{15}$$

$$\leq \sum_{u_1,\ldots,u_{i-1}} \Pr\left[u_1 \in U, \ldots, u_{i-1} \in U, u - \sum_{j=1}^{i-1} u_j \in U\right] \tag{16}$$

$$\approx \sum_{u_1,\ldots,u_{i-1}} \Pr\left[u_1 \in U\right] \ldots \Pr\left[u - \sum_{j=1}^{i-1} u_j \in U\right] \tag{17}$$

$$\approx \Pr[U * \ldots * U = u] \;/\; i!. \tag{18}$$

Equation (15) is a decomposition of events from equation (14). We use the union bound to obtain equation (16). In equation (17), we treat draws from U as independent draws, and write different draws as a product of probabilities. This can be approximated by convolution of independent copies of U divide by $i!$, which accounts for the number of times combinations of $u_1, \ldots, u_i - 1, u - \sum_{j=1}^{i-1} u_j$ have been double-counted. If the numerical value of $\Pr[u \in U_i]$ is greater than 1, it is set to 1. □

Collision probability and uniqueness of solution. Suppose we have more than one solution by the end of $\mathcal{A}_1$. This means that there is at least one non-genuine solution satisfying all the constraints. The solution has to be different from the original database by at least one volume. With some simplification, we assume that this volume is in the middle of the tuple, then the solution must satisfy two two-way out-of-order sum of observed volumes, three three-way out-of-order sum of observed volumes and so on. In other words, there are two volumes drawn from U_2 that are in U by chance, and so on.

We define k-way collision probability by

$$\Pr[k\text{-way collision}] = \max_x \left[\Pr[U(x) > 0] \Pr[x \in U_k]\right]. \tag{19}$$

This is the maximum probability that a random sum of k out-of-order observed volumes equal to one of the observed volumes. This allows us to derive the probability that the solutions of length b contains only genuine solutions.

Lemma 3. *Let $\tilde{V} = (V_1, \ldots, V_N)$ be the distribution of elementary volumes and b be the maximum window size. Let $\mathcal{S}_b$ be the solutions of length b. Then $\mathcal{S}_b$ contains only genuine solutions with probability approximately*

$$1 - \sum_{j=1}^{i-1} \binom{i-1}{j} |W|^j \prod_{k=2}^{i} \Pr[k\text{-way collision}]^{\min\{j \cdot k, i-k+1\}}.$$

Proof. Since $\left|\tilde{V}\right| = N$ and the maximum window size is b, there are at most $|W| = \sum_{i=0}^{b-1} N - i = bN - (b-1)b/2$ observed volumes. This means there are at most $b-1$ volumes that can differ from a genuine solution. Therefore, by computing the probability that there are $1 \leq i \leq b-1$ volumes differing from a genuine solution, we obtain a bound on the probability that $\mathcal{S}_b$ contains only genuine solutions.

In general, we can compute the probability that there are i volumes differ from a genuine solution as

$$\begin{aligned}&\Pr[i \text{ volumes are different from a genuine solution}]\\ \approx&\binom{b-1}{i}|W|^i \prod_{k=2}^{b} \Pr[k\text{-way collision}]^{\min\{i\cdot k, b-k+1\}}.\end{aligned}$$

In the equation, $\binom{b-1}{i}|W|^i$ accounts for the orientations of genuine volumes and wrong volumes, and the later product is an upper bound on the probability that a solution with i non-genuine volumes is valid, assuming independence of collisions.

Using union bound, we find the probability that the solution set contains only genuine solutions as

$$\begin{aligned}&\Pr[\mathcal{S}_b \text{ contains only genuine solutions}]\\ \approx&1 - \sum_{i=1}^{b-1} \Pr[i \text{ volumes are different from a genuine solution}]\\ \approx&1 - \sum_{i=1}^{b-1} \binom{b-1}{i}|W|^i \prod_{k=2}^{b} \Pr[k\text{-way collision}]^{\min\{i\cdot k, b-k+1\}}.\end{aligned}$$

□

The next Theorem establishes an estimation for p_{unique}.

Theorem 8 (Estimation of p_{unique})**.** *Let $\tilde{V} = (V_1, \ldots, V_N)$ be the distribution of elementary volumes and b be the maximum window size. We can estimate p_{unique} as*

$$\Pr[\mathcal{S}_b \textit{ contains only genuine solutions}] - (N-b)q,$$

where $q = 2|W| \prod_{k=2}^{b+1} \Pr[k\textit{-way collision}]$.

Proof. We know from Lemma 3 there is a certain probability that the solutions of length b contains only genuine solutions. We seek the probability that the solution set remains genuine for all further iterations. There are $|W|$ to be tested on the left and right of the genuine solutions in each iteration. If our algorithm finds a solution by chance, there must be a k-way collision for all $2 \leq k \leq b$.

Hence, using union bound, we get

$$\begin{aligned}
&\Pr\left[\begin{smallmatrix}\mathcal{S}_{j+1}\text{ contains}\\ \text{only genuine solutions}\end{smallmatrix} \mid \begin{smallmatrix}\mathcal{S}_{j}\text{ contains}\\ \text{only genuine solutions}\end{smallmatrix}\right]\\
\approx &1 - 2|W| \prod_{k=2}^{b} \Pr[k\text{-way collision}]\\
= &1 - q.
\end{aligned}$$

Applying union bound again, we find

$$\begin{aligned}
&p_{unique}\\
\approx &\Pr[\mathcal{S}_b \text{ contains only genuine solutions}]\\
&- \sum_{j=b+1}^{N} 1 - \Pr\left[\begin{smallmatrix}\mathcal{S}_{j+1}\text{ contains}\\ \text{only genuine solutions}\end{smallmatrix} \mid \begin{smallmatrix}\mathcal{S}_{j}\text{ contains}\\ \text{only genuine solutions}\end{smallmatrix}\right]\\
= &\Pr[\mathcal{S}_b \text{ contains only genuine solutions}] - (N-b)q.
\end{aligned}$$

□

B.3 Theoretical Analysis of uniqueness and Simulation

In this section we compare the quality of the theoretical bound on uniqueness with experimental simulation. We perform experiments on distributions of databases where the elementary volumes are modelled by independent and identically distributed binomial distributions, for $b = 5$ to 8, and $N = 40$. There are two types of synthetic distributions we have considered, namely databases with an increasing/decreasing elementary volumes and those with an inverted-U shape.

For the first type of database, we sample the elementary volumes as $\mathbf{v}_i \sim \text{Binom}(20i + 200, 0.5)$. For the second type of database, we sample the elementary volumes as $\mathbf{v}_i \sim \text{Binom}(400 - 20i, 0.5)$ for $N \leq 20$ and $\mathbf{v}_i \sim \text{Binom}(20i - 200, 0.5)$ for $N > 20$. We compute the theoretical estimations of the uniqueness rates for the two types of databases and compare those to the simulations.

For the simulations, we generate 1000 databases from the type of database in interest and execute our basic attack. We report the uniqueness rates with out the final leakage check procedure (line 7 of Algorithm 2) so that the comparison with our theoretical estimations is more direct. The experimental results are shown in Table 12 respectively.

C Simple Variations on the Leakage

C.1 Correctness of Attack with Noise

Proof of Theorem 2. Statement (2) follows immediately if we can prove statement (1). To prove statement (1), we use induction to show that $\forall \mathbf{v}' \in W^i, \mathcal{L}_b(\mathbf{v}') \subseteq W \Rightarrow \mathbf{v}' \in \mathcal{S}_i$. The proof is identical to that of the basic attack. □

b	Experimental	Theoretical
5	0.0%	98.1%
6	73.8%	98.1%
7	95.8%	99.8%
8	99.5%	99.7%

b	Experimental	Theoretical
5	0.0%	85.9%
6	0.0%	98.7%
7	40.4%	99.7%
8	88.0%	99.6%

Table 12: Experiments on the databases with a decreasing shape (left) and inverted-U shape (right).

C.2 Pseudocode of the Attack with Missing Queries

Full pseudocode of our attack $\mathcal{A}_3$ is shown in Algorithm 4. The main structure of the attack is quite similar to that of the basic attack, so we only highlight the key differences between the two attacks. In lines 5 and 6 of the algorithm, we extend the solutions in a way that is quite similar to that in the basic attack. The difference is that we need to check the number of missing constraints, as of line 12 to 15 of the algorithm, and eliminate the solutions with too many missing constraints in line 17. All the solutions of length N are returned by the algorithm.

C.3 Correctness of the Attack with Missing Queries.

Proof of Theorem 3. Statement (2) comes for free if we can prove statement (1), so it suffices to prove the first statement.

($\Rightarrow$) Assume that $\mathbf{v}'$ is a solution that satisfies $\mathcal{L}_{b,\mathcal{I}'}(\mathbf{v}') \subseteq W$ with $\mathcal{I}'$ satisfying the same constraints as $\mathcal{I}$. We show that all solutions of this form are in $\mathcal{S}_N$.

We prove with induction that in each iteration, $\forall \mathbf{v}' \in W^i, \forall \mathcal{I}' \in ([N] \times [N])^*, \mathcal{L}_{b,\mathcal{I}'}(\mathbf{v}') \subseteq W \Rightarrow \mathbf{v}' \in \mathcal{S}_i$. The base case is trivial as the minimum volume satisfies all the constraints. Now we assume that the statement is true for some iteration i. We assume without loss of generality that $(w_1, \ldots, w_i)$ is one of the solutions in $\mathcal{S}_i$. Our algorithm in line 12 to 17 checks if some new volume w_0 can satisfy sufficiently many constraints between indices 0 and $b-1$. We add $(w_0, \ldots, w_i)$ to the solution set $\mathcal{S}_{i+1}$ if no more than k constraints are missing. We do not need to check the constraints between other windows as they have already been checked in the previous iterations, as stated by our assumption. Therefore, we conclude that the partial solutions satisfies sufficiently many constraints during each iteration and the proof is complete. □

C.4 Pseudocode of the Attack with Padded Queries

Pseudocode of our attack $\mathcal{A}_4$ is shown in Algorithm 5. R in our attack is a set of pairs of volumes which the adversary computes as a guess of the ranges of true volumes. Each element of R is of the form (x, y), where x is the lower bound and y is the upper bound on the guess. We do not specify how the adversary should guess the true volumes in general as it depends on the distribution of fake records

in general. In the toy version where the number of fake records for a query follows a discrete uniform distribution between 1 and r, and it is not changed for further queries, we can compute R trivially as $\{(w-r, w-1) \mid w \in W\}$. The adversary can make less accurate guesses on the true observed volumes as an attempt to achieve unique database reconstruction.

The adversary initialise $\mathcal{S}_1$ as the pair of volumes containing the minimum observed volume. In the pseudocode, we abuse $\min(R)$ to mean the operation described above. Solutions are extended in line 5 to 6 just like the basic attack. Unlike the equality constraints in the basic attack, presence of fake records means the adversary is only able to check if the solutions can be padded to generate the same observed volumes as he is given. Finally, some of the solutions are eliminated in line 7 to 9 by checking if all padded volumes can be derived from the solutions.

C.5 Correctness of the Attack with Padded Queries

Proof of Theorem 4. Statement (2) comes for free if we can prove statement (1), so it suffices to prove the first statement.

($\Leftarrow$) We get this implication for free as line 7 to 9 of the algorithm ensures the condition.

($\Rightarrow$) Suppose $\mathbf{v}'$ can generate the set of observed volumes, i.e. $W \in \text{supp}(\mathcal{L}_{b,Z}(\mathbf{v}'))$, we show that an estimation of the solution is in the final solution set. We use induction to show that the support of the leakage of the partial solutions are subsets of W. As the algorithm checks for the equality in line 7 to 9, we obtain the desired result in the end.

We verify that in the base case when a subset of W is in the support of the leakage of the partial solution. This is indeed the case as the smallest entry in R includes the minimal volume in the real database. In the induction step, we assume that $\mathbf{v}'$ is a database that can generate a subset of W, then there is a solution $(s[1], \ldots, s[i])$ such that $\forall i, s[i][1] \leq \mathbf{v}' \leq s[i][2]$. Without loss of generality, we look at the case where the solution is extended to the left. Let w be some volume that can be used to extend $\mathbf{v}'$ such that the extended database can generate a subset of W, then there must be a pair of volumes $(x, y) \in R$ such that $x \leq w \leq y$. The condition check in line 15 of the algorithm must succeed on (x, y) by the definition of the leakage function. So the induction step is complete. □

D Attack on Observed Volumes with Bounded Window Size

D.1 Attack on Bounded Window Sizes

Full pseudocode of our attack $\mathcal{A}_5$ on observed volumes with minimum and maximum window sizes is shown in Algorithm 6.

D.2 Correctness of Attack on Bounded Window Sizes

We prove the correctness of the sub-routines one by one, and then establish the correctness of the overall algorithm. At a high level, the correctness of the sub-routines states that if we begin with a database that has the same leakage profile as what is given, and run the sub-routines with appropriate inputs including some transformation(s) of the database, then the outputs contain some other transformation(s) of the database. To prove that our attack is correct as a whole, it suffices to show that all the sub-routines output the expected results. For readability, we write $\mathbf{v}[i]$ to mean i-th entry of $\mathbf{v}$, and $\mathbf{v}[i,j]$ to mean $\sum_{k=i}^{j} \mathbf{v}[k]$. All proofs refer to pseudo-code shown in Algorithm 6.

Lemma 4 (Correctness of the Initial Solution)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* a, b *be natural numbers less or equal to* N *with* $b > 2a$*. Let* $\mathcal{S} = \mathcal{A}_2(\mathcal{L}_{a,b}(\mathbf{v}), b/a, N/a)$*. Let* $s = (\mathbf{v}[1,a], \mathbf{v}[a+1, 2a], \ldots, \mathbf{v}[N-a+1, N])$*. Then*

$$(s \in \mathcal{S}) \vee \left(s^R \in \mathcal{S}\right).$$

Proof. The correctness of the initial solution is a direct consequence of the correctness of z with all observed volumes used as the initial solution. □

Lemma 5 (Correctness of procedure Offset Solutions)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* a, b *be natural numbers less or equal to* N *with* $b > 2a$*. Let* $s = (\mathbf{v}[1,a], \mathbf{v}[a+1, 2a], \ldots, \mathbf{v}[N-a+1, N])$*. Let* $\mathcal{S} =$ Offset Solutions$(s, \mathcal{L}_{a,b}(\mathbf{v}), b/a, N/a)$*. Then for all* $s_k \coloneqq (\mathbf{v}[1, a+k], \mathbf{v}[a+k+1, 2a+k], \ldots, \mathbf{v}[N-2a+k+1, N])$ *with* $1 \le k < a$*, we have*

$$s_k \in \mathcal{S}.$$

Proof. We prove the statement for an arbitrary k. In line 23 to 26, we find the initial set of solutions by checking conditions in line 25. It is not hard to see that $\mathbf{v}[1, a+k]$ satisfies all the conditions, so $(\mathbf{v}[1, a+k]) \in \mathcal{S}_1$. By construction, in iteration i, $\mathbf{v}[(i-1)a+k+1, ia+k]$ satisfies all constraints specified in line 31, so $\mathcal{S}_i$ contains first i volumes of s_k (as a tuple) for all $i \le N/a - 2$. Finally, line 33 to 36 append last volumes to the solutions, if the resulting solution is still plausible. s_k is a partial solution with correct last volume, so $s_k \in \mathcal{S}_{N/a}$. □

Lemma 6 (Correctness of procedure Merge Solutions)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* a, b *be natural numbers less or equal to* N *with* $b > 2a$*. Let* $s = (\mathbf{v}[1,a], \mathbf{v}[a+1, 2a], \ldots, \mathbf{v}[N-a+1, N])$*. Define* $s_k \coloneqq (\mathbf{v}[1, a+k], \mathbf{v}[a+k+1, 2a+k], \ldots, \mathbf{v}[N-2a+k+1, N])$ *with* $1 \le k < a$*. Let* $\mathcal{S} =$ Merge Solutions$(\mathcal{S}', s, \mathcal{L}_{a,b}(\mathbf{v}), a, b, N)$*. If* $s_k \in \mathcal{S}'$ *for all* k*, then*

$$(\mathbf{v}[1,a], \mathbf{v}[a+1], \ldots, \mathbf{v}[N-a], \mathbf{v}[N-a+1, N]) \in \mathcal{S}.$$

Proof. We show that line 39 to 47 produce the accumulated version of the desired result. By that, we mean the volumes in the middle are of the form $\mathbf{v}[1, j]$ as opposed to $\mathbf{v}[j]$. It suffices to show that at iteration i, $(\mathbf{v}[1,a], \mathbf{v}[1, a+1], \ldots, \mathbf{v}[1, (i+1)a-1]) \in \mathcal{S}_i$. We prove this with induction.

At iteration $i = 1$, we start building the set of solutions with $\mathbf{v}[1, a]$ in line 41. In line 42, we know that $\mathbf{v}[1, ia+k] \in X$ for all $k = 1, \ldots, a-1$, as $s_k \in \mathcal{S}'$ for all k. As a result, $(\mathbf{v}[1, a], \mathbf{v}[1, a+1], \ldots, \mathbf{v}[1, 2a-1]) \in \mathcal{S}_1$ because all conditions in line 45 and 47 are satisfied. Inductive step works in the same way as the base case, and we conclude that $(\mathbf{v}[1, a], \mathbf{v}[1, a+1], \ldots, \mathbf{v}[1, N-a-1]) \in \mathcal{S}_{N/a-2}$.

In line 48, the last two volumes are added to each of the solutions in $\mathcal{S}_{N/a-2}$, and all the volumes except the first and the last volumes are transformed to elementary volumes. By the correctness of the operation, we obtain the desired solution. □

Lemma 7 (Correctness of procedure Finalise Solution)**.** *Let* $\mathbf{v}$ *be a database,* $N = |\mathbf{v}|$ *and* a, b *be natural numbers less or equal to* N *with* $b > 2a$*. Let* $s = (\mathbf{v}[1, a], \mathbf{v}[a+1], \ldots, \mathbf{v}[N-a], \mathbf{v}[N-a+1, N])$*. Let* $\mathcal{S} =$ Finalise Solutions$(s, \mathcal{L}_{a,b}(\mathbf{v}), a, b, N)$*. Then*

$$\mathbf{v} \in \mathcal{S}.$$

Proof. In line 53, $\mathbf{v}[1, a]$ and $\mathbf{v}[N-a+1, N]$ are removed from s. In line 54 to 60, we try to extend s to the left by finding volumes that look like $\mathbf{v}[1, a], \ldots, \mathbf{v}[a-1, 2a-1]$. The real database $\mathbf{v}$ with last a volumes removed satisfies all the conditions in line 58 by default. Using the same argument to extend the solution to the right, we conclude that $\mathbf{v} \in \mathcal{S}$. □

We are ready to prove our main theorem.

Proof of Theorem 5. We get statement (2) for free if we can prove statement (1). Backward implication of statement (1) is trivial, as equality of leakage is checked in line 12 of the attack. It remains to prove the forward implication.

($\Rightarrow$) Let $\mathbf{v}'$ be any database with $\mathcal{L}_{a,b}(\mathbf{v}') = \mathcal{L}_{a,b}(\mathbf{v})$, we show that $\mathbf{v}'$ is one of the solutions to $\mathcal{A}_5(\mathcal{L}_{a,b}(\mathbf{v}), a, b, N)$. By the correctness of the procedure Initial Solution, we know that $s = (\mathbf{v}'[1, a], \mathbf{v}'[a+1, 2a+1], \ldots, \mathbf{v}'[N-a+1, N])$ or its reflection is one of the solutions returned by procedure Initial Solution. Without loss of generality, we assume s is one of the solutions. By the correctness of the procedure Offset Solutions, the offset solutions s_k associated to $\mathbf{v}'$ are contained in the set of solutions returned by Offset Solutions. This means the procedure Merge Solutions returns solutions including $\mathbf{v}'$ with volumes in range 1 to a and $N-a+1$ to N merged. Finally, with the correctness of the procedure Finalise Solution, the solution found in the previous step is restored to $\mathbf{v}'$ and potentially some other solutions. Check of equality of leakage in line 12 does not affect $\mathbf{v}'$ as we assumed $\mathcal{L}_{a,b}(\mathbf{v}') = \mathcal{L}_{a,b}(\mathbf{v})$ from the beginning. Therefore, we conclude that $\mathbf{v}' \in \mathcal{A}_5(\mathcal{L}_{a,b}(\mathbf{v}), a, b, N)$ and the proof is complete. □

Algorithm 4 Attack with missing queries

input $W = \{\sum_{i=x}^{y} \mathbf{v}_i \mid (x,y) \in \mathcal{I}, y - x + 1 \leq b\}, b, k, N$
output $\{(w_1, \ldots, w_N) \mid w_i \in W\}$

procedure ATTACK(W, b, k, N)
    $\mathcal{S}_1 = \{(\min(W))\}$
    for $i \leftarrow 2, N$ **do**
        $\mathcal{S}_i \leftarrow$ EXTENDLEFT($\mathcal{S}_{i-1}, W, b, k$)$\cup$
         EXTENDRIGHT($\mathcal{S}_{i-1}, W, b, k$)
    return $\mathcal{S}_N$

procedure EXTEND LEFT($\mathcal{S}_i, W, b, k$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_1, \ldots, w_i) \in \mathcal{S}_i$ **do**
        for all $w_0 \in W$ **do**
            $m \leftarrow 0$
            for all $(x,y), x < y, y \leq b - 1$ **do**
                if $\left(\sum_{j=x}^{y} w_j\right) \notin W$ **then**
                $m \leftarrow m + 1$
            if $m \leq k$ **then**
                $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w_0, w_1, \ldots, w_i)\}$
    return $\mathcal{S}'$

procedure EXTEND RIGHT($\mathcal{S}_i, W, b, k$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_1, \ldots, w_i) \in \mathcal{S}_i$ **do**
        for all $w_{i+1} \in W$ **do**
            $m \leftarrow 0$
            for all $(x,y), x < y, x \geq i - b + 1$ **do**
                if $\left(\sum_{j=x}^{y} w_j\right) \notin W$ **then**
                $m \leftarrow m + 1$
            if $m \leq k$ **then**
                $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w_1, \ldots, w_i, w)\}$
    return $\mathcal{S}'$

Algorithm 5 Attack with padded queries

input $R, W = \{\sum_{i=x}^{y} \mathbf{v}_i + r \mid y - x + 1 \leq b, r \leftarrow Z\}, b, N, r = \max(Z)$
output $\{(w_1, w_2), \ldots, (w_{2N-1}, w_{2N}))\}$

procedure ATTACK(R, W, b, N, r)
    $\mathcal{S}_1 = \{(\min(R))\}$
    for $i \leftarrow 2, N$ **do**
        $\mathcal{S}_i \leftarrow$ EXTENDLEFT($R, \mathcal{S}_{i-1}, W, b, r$)$\cup$
        EXTENDRIGHT($R, \mathcal{S}_{i-1}, W, b, r$)
    for all $s \in \mathcal{S}_N$ **do**
        if $\exists w \in W, \forall (x, y), \sum_{i=x}^{y} s[i][1] < w - r$ and $\sum_{i=x}^{y} s[i][2] > w - 1$ **then**
            $\mathcal{S}_N \leftarrow \mathcal{S}_N \setminus \{s\}$
    return $\mathcal{S}_N$

procedure EXTEND LEFT($\mathcal{S}_i, R, W, b, r$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_1, \ldots, w_i) \in \mathcal{S}_i$ **do**
        for all $w_0 \in R$ **do**
            if $\forall j < b, \exists w \in W, \sum_{k=0}^{j} w_k[1] \geq w - r$ and $\sum k = 0^j w_k[2] \leq w - 1$ **then**
                $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w_0, w_1, \ldots, w_i)\}$
    return $\mathcal{S}'$

procedure EXTEND RIGHT($\mathcal{S}_i, R, W, b, r$)
    $\mathcal{S}' \leftarrow \{\}$
    for all $(w_i, \ldots, w_1) \in \mathcal{S}_i$ **do**
        for all $w_0 \in R$ **do**
            if $\forall j < b, \exists w \in W, \sum_{k=0}^{j} w_k[1] \geq w - r$ and $\sum k = 0^j w_k[2] \leq w - 1$ **then**
                $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{(w_i, \ldots, w_1, w_0)\}$
    return $\mathcal{S}'$

Algorithm 6 Attack with bounded window size

input $W = \{\sum_{i=x}^{y} v_i \mid a \leq y - x + 1 \leq b\}, a, b, N$
output $\{(w_1, \ldots, w_N) \mid w_i \in W\}$

procedure ATTACK(W, b, N)
    $X \leftarrow \emptyset$
    $\mathcal{S} \leftarrow$ INITIAL SOLUTION($W, b/a, N/a$)
    for all $s \in \mathcal{S}$ **do**
        $\mathcal{S}' \leftarrow$ OFFSET SOLUTIONS($s, W, b/a, N/a$)
        $\mathcal{S}' \leftarrow$ MERGE SOLUTIONS($\mathcal{S}', s, W, a, b, N$)
        for all $s' \in \mathcal{S}'$ **do**
            $s' \leftarrow$ FINALISE SOLUTION(s', W, a, b, N)
            $X \leftarrow X \cup \{s'\}$
    $X \leftarrow \{x \mid x \in X, \mathcal{L}_{a,b}(x) = W\}$
    return X

procedure INITIAL SOLUTION(W, b, N)
    $\mathcal{S}_1 = \{(w) \mid \ w \in W\}$
    for $i \leftarrow 2, N$ **do**
        $\mathcal{S}_i \leftarrow \text{EXTENDLEFT}(\mathcal{S}_{i-1}, W, b) \cup \text{EXTENDRIGHT}(\mathcal{S}_{i-1}, W, b)$
    for $s \in \mathcal{S}_N$ **do**
        if $s^R \in \mathcal{S}_N$ **then**
            $\mathcal{S}_N \leftarrow \mathcal{S}_N \backslash \{s\}$
    return $\mathcal{S}_N$

procedure OFFSET SOLUTIONS(s, W, b, N)
    $\mathcal{S}_1 \leftarrow \emptyset$
    for all $w \in W$ **do**
        if $(w > s[1]) \quad \wedge \quad (w < s[1] + s[2]) \quad \wedge \quad \left\{\left(\sum_{j=1}^{k} s[j]\right) - w \mid k = 3, \ldots, b+1\right\} \subset W$ **then**
            $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{(w)\}$
    for all $i \leftarrow 2, N-2$ **do**
        $\mathcal{S}_i = \emptyset$
        for all $s' \in \mathcal{S}_{i-1}$ **do**
            for all $w \in W$ **do**
                if $\left(w + \sum_j s'[j] > \sum_{j=1}^{i} s[j]\right) \wedge \left(w + \sum_j s'[j] < \sum_{j=1}^{i+1} s[j]\right) \wedge \left\{w + \sum_{j=k}^{i-1} s'[j] \mid k = \max\{1, i-b+1\}\right\} \subset W \wedge \left\{\sum_{j=0}^{k} s[j] - \sum_j s'[j] - w \mid \ k = i+2, \ldots, \min\{|s|, i+b+1\}\right\} \subset W$ **then**
                    $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{s' + (w)\}$
    $\mathcal{S}_{N-1} = \emptyset$
    for all $s' \in \mathcal{S}_{N-1}$ **do**
        if $\left\{\sum_i s[i] - \sum_{i=1}^{j} s'[i] \mid \ j = N-b, \ldots, N-2\right\} \subset W$ **then**
            $\mathcal{S}_{N-1} \leftarrow \mathcal{S}_{N-1} \cup \{s' + (\sum_i s[i] - \sum_i s'[i])\}$
    return $\mathcal{S}_{N-1}$

procedure MERGE SOLUTIONS($\mathcal{S}, s, W, a, b, N$)
  $\mathcal{S}_0 \leftarrow \{()\}$
  for all $i \leftarrow 1, N/a - 2$ **do**
    $\mathcal{S}_i \leftarrow \left\{ s' + \left(\sum_{j=1}^{i} s[j] \right) \mid s' \in \mathcal{S}_{i-1} \right\}$
    $X \leftarrow \left\{ \sum_{j=1}^{i} s'[j] \mid s' \in \mathcal{S} \right\}$
    for all $x \in X$ **do**
      for all $s' \in \mathcal{S}_i$ **do**
        if $\left\{ x + \sum_{j=a-1}^{b} s'[j] \right\} \subset W \wedge \left\{ \left(\sum_{j=1}^{k} s[j] \right) - x \mid k = i+2, \ldots, i+b/a+1, k < |s| \right\} \subset W$ **then**
          $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{s' + (x)\}$
    $\mathcal{S}_i \leftarrow \{s' \mid s' \in \mathcal{S}', |s'| = i \cdot a\}$
  $\mathcal{S}' \leftarrow \{(s'[0], s'[1] - s'[0], \ldots, s'[N-a] - s'[N-a-1], \sum_i s[i] - s[N/a - 1] - s'[N-a], s[N/a-1] - s[N/a]) \mid s' \in \mathcal{S}_{N/a-2}\}$
  return $\mathcal{S}'$

procedure FINALISE SOLUTION(s, W, a, b, N)
  if $s = ()$ **then**
    return $\emptyset$
  $\mathcal{S}' = \{s[2 : |s| - 1]\}$
  for all $i \leftarrow 1, a$ **do**
    $\mathcal{S}_{tmp} \leftarrow \emptyset$
    for all $x \in \mathcal{S}'$ **do**
      for all $w \in W$ **do**
        if $\left\{ w + \sum_{j=a}^{k} x[j] \mid \ k = a, \ldots, b-1 \right\} \subset W$ **then**
          $\mathcal{S}_{tmp} \leftarrow \mathcal{S}_{tmp} \cup \left\{ \left(w - \sum_{j=1}^{a-1} x[j] \right) + x \right\}$
    $\mathcal{S}' = \mathcal{S}_{tmp}$
  $\mathcal{S}' \leftarrow \{x^R \mid \ x \in \mathcal{S}'\}$
  for all $i \leftarrow 1, a$ **do**
    $\mathcal{S}_{tmp} \leftarrow \emptyset$
    for all $x \in \mathcal{S}'$ **do**
      for all $w \in W$ **do**
        if $\left\{ w + \sum_{j=k}^{|x|-a+1} x[j] \mid \ k = |x| - b, \ldots, |x| - a + 1 \right\} \subset W$ **then**
          $\mathcal{S}_{tmp} \leftarrow \mathcal{S}_{tmp} \cup \left\{ x + \left(w - \sum_{j=0}^{a-2} x[|x| - j] \right) \right\}$
    $\mathcal{S}' = \mathcal{S}_{tmp}$
  return $\mathcal{S}'$