Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and Information Systems

School of Computing and Information Systems

3-2022

Update recovery attacks on encrypted database within two updates using range queries leakage

Jianting NING
Singapore Management University, jtning@smu.edu.sg

Geong Sen POH
NUS-Singtel Cyber Security Research and Development Laboratory

Xinyi HUANG
Fujian Normal University

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Shuwei CAO
NUS-Singtel Cyber Security Research and Development Laboratory

See next page for additional authors

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Information Security Commons

Citation

NING, Jianting; POH, Geong Sen; HUANG, Xinyi; DENG, Robert H.; CAO, Shuwei; and CHANG, Ee-Chien. Update recovery attacks on encrypted database within two updates using range queries leakage. (2022). *IEEE Transactions on Dependable and Secure Computing*. 19, (2), 1164-1180.
Available at: https://ink.library.smu.edu.sg/sis_research/5998

This Journal Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylds@smu.edu.sg.

Author
Jianting NING, Geong Sen POH, Xinyi HUANG, Robert H. DENG, Shuwei CAO, and Ee-Chien CHANG

This journal article is available at Institutional Knowledge at Singapore Management University:
https://ink.library.smu.edu.sg/sis_research/5998

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

Update Recovery Attacks on Encrypted Database within Two Updates using Range Queries Leakage

Jianting Ning, Geong Sen Poh, Xinyi Huang, Robert H. Deng, *Fellow, IEEE*, Shuwei Cao, and Ee-Chien Chang

Abstract—Recently, reconstruction attacks on static encrypted database supporting range queries have been proposed. However, attacks on encrypted database within two updates in the similar setting have not been studied extensively. As far as we know, the only work is the *update recovery attack* presented by Grubbs *et al.* (CCS 2018). Following their seminal work, we present new update recovery attacks for *dense* dataset (i.e. at least one record corresponding to each value in the range), which enable a deeper understanding of the impact caused by leakages due to updates on dynamic encrypted database. Our first attack aims at recovering the value of a newly added record in the case of one database update. We further demonstrate that the attack can fully reconstruct the *database counts* if the updated value is either the minimum or maximum in the range. We then consider a setting where two distinct records are added separately, which leads to our second attack. We next extend our attacks to the setting where the update operation is deletion. To the best of our knowledge, update recovery attack on database supporting deletion has not been considered before. We demonstrate practicality of our attack via extensive simulations using real dataset.

Index Terms—Dynamic encrypted database, update recovery attacks, range query, information leakage.

1 Introduction

In recent years, novel and practical attacks on secure outsourced database based on *volume leakage* have been proposed. In these attacks, an attacker learns only the number of records returned from a range query. Kellaris *et al.* [23], in a seminal work, introduced and demonstrated such attack on secure outsourced database supporting range queries. In their attack, the exact number of records for each value in the range, or termed as the *database counts* [15], can be recovered. This is in contrast to many existing works, where the attacker is assumed to have knowledge of the *access pattern leakage [24]* or reference distribution of the database [18]. Most recently, Grubbs *et al.* [15] improved the work of Kellaris *et al.* by presenting a more practical *database reconstruction attack* for recovering the database counts. Besides their main result, a new attack termed *update recovery attack* is presented. The setting is such that given the knowledge of the reconstructed database counts, the attacker is able to approximate the value of a new record added to the database.

Attacks based on volume leakage can be mounted quite easily and may have serious consequences as were stated in [15], [23]. For instance, in the case whereby the server is compromised for a period of time, the attacker will be able to learn the volume of each range query. The same reasoning is applicable if the server is the attacker itself, even when the outsourced database deploys advanced techniques that hide other auxiliary information. By learning database counts, an attacker may be able to learn the underlying value of an added record. We refer reader to [15], [23] for additional, detailed discussions and motivation on attacks based on volume leakage.

Existing work. As mentioned above, Kellaris *et al.* [23] constructed the first attack using volume leakage on secure outsourced database supporting range queries. However, as stated in [15], Kellaris *et al.*'s attack can only work in the setting where the range queries are drawn at random in an independent and uniform manner. Furthermore, their attack requires observation of $\mathcal{O}(N^4 logN)$ queries, where N is the maximum value in the range. These limit practicality of their attack. Aiming at a more practical attack, Grubbs *et al.* [15] (GLMP) presented a new novel database reconstruction attack for range queries.

The series of recent attacks [15], [23] demonstrated the devastating effect of volume leakage on *static* encrypted database supporting range queries, but the effect of these attacks on *dynamic* setting has not been fully studied. As far as we know, the only work targeted on dynamic database is the update recovery attack proposed by GLMP [15], which aims to recover the value of a newly added record. For the

- *J. Ning is with Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China, and the School of Information Systems, Singapore Management University, Singapore.*
  E-mail: jtning88@gmail.com
- *G. S. Poh and S. Cao are with NUS-Singtel Cyber Security Research and Development Laboratory, Singapore.*
  E-mail: geongsen@gmail.com, dcscaos@nus.edu.sg.
- *X. Huang is with Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China.*
  E-mail: xyhuang81@gmail.com.
- *R. H. Deng is with the Secure Mobile Centre, School of Information Systems, Singapore Management University, Singapore.*
  E-mail: robertdeng@smu.edu.sg.
- *E. Chang is with the Department of Computer Science, National University of Singapore, Singapore.*
  E-mail: changec@comp.nus.edu.sg.

*Manuscript received *; revised *.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

attack to succeed, they assume that the attacker knows the exact database counts, which can only be achieved assuming the database is *dense* in the sense that there exists at least one record corresponding to each value in the range. Given the knowledge of the database counts, their update recovery attack adopts a probabilistic algorithm that approximate the value of newly added record for general setting. In the event of record deletion during an update operation, it is not known how their attack would work, especially for the case when the database becomes non-dense after deletion. Taking the above limitations into consideration, two questions arise naturally:

- *Is there any approach that can launch the update recovery attack directly (i.e., without first performing the complex reconstruction of the database counts), and that could deterministically recover the value of the added record (up to reflection*[1]*)?*
- *Does there exist any update recovery attack on dynamic encrypted database supporting deletion?*

We believe it is crucial to investigate the effect of update since for secure outsourced database, adding or deleting a record are routine operations. Because of database update, more information may be leaked. The consequences due to such leakage should be carefully studied in order to understand what type of attacks a dynamic encrypted database will encounter. This is what motivates us in this work.

Our results. We develop two new update recovery attacks for dynamic encrypted database supporting range queries that could recover the value of newly updated record. Following the two update recovery attacks, we present their extensions that work in the case where the update operation is deletion. As far as we know, these are the first attacks against dynamic encrypted database supporting deletion. Our work highlights the security impact of information leakage incurred by the update operations over dynamic encrypted database supporting range queries. All our attacks operate under a rather weak passive adversarial model, in which we do not make any assumption on the query or data distribution. What is needed, is just the observation that covers the volume and *access location* of each range query at least once. Access location, which we explain in Section 2.2, is a weaker source of leakage than *access pattern*.

In more details, we study the problem of update recovery attack for dynamic encrypted database given only leakages of volume and access location. Our main results are summarized as below:

- We present a new update recovery attack for the scenario where one record is added. Our attack can recover the value of the added record directly (i.e., without first recovering the database counts as in [15]), up to reflection. This attack stems from several observations we present. Based on our observations, we can further recover the database counts. In particular, for the case when the value of the added record is either the minimum or the maximum value in the range, we can fully reconstruct the database counts.
- We consider a more generic scenario where two different records are added separately. Again, we present a series of observations for this setting. Stems from the observations, the attack can recover the values of the two added records, up to reflection. We can further recover the database counts from our observations. In particular, for the case when the values are consecutive numbers or one of the values is either the minimum or the maximum value in the range, the database counts can be fully reconstructed.
- We further present extensions to the above two attacks for the scenario where the update operation is deletion, in which the database may become non-dense after delection. As far as we know, these are the first attacks that consider this scenario.

Our attack outperforms the work of GLMP in terms of recovery precision. In particular, in the scenario where one record is added, our attack can deterministically recover the value of newly added record (up to reflection), while GLMP's attack is sometimes only able to approximate this value. This fact is reflected in our experiments for most of the attributes selected from the medical dataset that were tested. In the scenario where two different records are added, our attack can fully recover the updated values, while GLMP's attack does not consider this scenario. In addition, deletion is also not considered in GLMP's attack. All of our attacks require leakages of volume and access location, but do not need the knowledge of the maximum value in the range and the total number of records, when compared to GLMP's attack. Nevertheless, GLMP's attack can start with fewer range queries once the database counts are known, but with less recovery precision.

Overall, from the point of recovering the value of the added record, our attacks provide a more direct, efficient and general way when compared with GLMP's attack. All our attacks stem from a series of observations we present, some of which are non-trivial as they provide new insights into further understanding the leakage of dynamic encrypted database. Our work shows that volume leakage, as well as access location leakage, should be considered a serious concern in practice, for the dynamic setting of encrypted database. The observations we presented could serve as important guidelines for database encryption scheme developer. Our work also serves as a reminder for researchers to seriously consider update leakage when developing new database encryption schemes.

Limitation of attack. Our attack has several limitations. The first limitation is that our attacks require the collection of the *full-cover volume set* or *full-cover access-volume pair set* that covers each range query at least once (before and after the update). We note, however, this is the same setting as in GLMP. As stated in [15], this is a strong assumption, but a weaker one compared with Kellaris *et al.*'s attack [23]. The assumption of Kellaris *et al.*'s attack needs each range query to be collected multiple times. The second limitation is that our attacks will not work in the case where the database is non-dense before any update. Again, this is the same as in GLMP. In particular, in GLMP's attack, the exact volume of each value in $[N]$ must first be recovered, which only works when the database is dense. In addition, our attack cannot work in the following cases: (1) multiple tuples related to different existing values will be inserted in the dataset (i.e., in the case of bulk insertion), which includes the extreme

1. It means for any value r, the recovered value could be for r or $N+1-r$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

case where the exact number of records for each value in the range are the same; (2) insert and delete operation will be executed simultaneously; (3) insert operation will insert new values in the dataset that increase domain size (which is similar to the case of non-dense setting).

Countermeasures. The main idea of protecting dynamic encrypted database against our attacks is to prevent the attacker from obtaining the exact volume and access location information. In terms of volume leakage, the aim is to hide the volume information of each range query. As stated in [15], one can batch several range queries together to hide the individual volume. Another approach is to add noise in the sense that some dummy records will be returned for each range query. However, these will inevitably incur bandwidth overhead or server storage overhead. In addition, the "bucketed" method suggested in [15] can prevent our attack to some extent in that the actual counts of individual record are not revealed. In brief, this means records with values that are close to one another are grouped as one logical value. In terms of access location, one possible countermeasure is to add noise such that the access locations of range queries are the same. In this case, advanced cryptographic techniques (such as ORAMs [6], [13], [34], fully homomorphic encryption [12], [35], secret sharing [4], [5]) should be employed. As one of the countermeasures, Kellaris *et al.* [22] combines ORAM and differential privacy to prevent database reconstruction attacks. However, this will inevitably incurs additional bandwidth and response time costs [3], [30].

2 Background

2.1 Preliminaries

We first present the abstract model of encrypted outsourced database, which is based on the model in [23], [27].

An encrypted database is a collection of records associated with search keys: $\mathcal{E} = \{(R_1, k_1), ..., (R_l, k_l)\}$. All records are assumed to have fixed length λ, and the search keys are belong to a domain $\mathcal{K}$. In essence, we can view the search keys as the indexing information for the database. A query is a predicate $\mathcal{Q} : \mathcal{K} \to \{0, 1\}$. Executing a query $\mathcal{Q}$ to an encrypted database $\mathcal{E}$ results in all records satisfying $\mathcal{Q}(\mathcal{E}) = \{R_i : \mathcal{Q}(k_i) = 1\}$. An encrypted outsourced database system for a collection of queries Q consists of the following two protocols between a client and a server. The first is the setup protocol, where the input of the client is $\mathcal{E} = \{(R_1, k_1), ..., (R_l, k_l)\}$ and the server has no input. After the setup protocol, the output of the client is a key K for query and the output of the server is a data structure $\mathcal{S}$. The second is the query protocol, where the input of the server is $\mathcal{S}$ and the input of the client is a query $\mathcal{Q} \in Q$ and the key K. After the query protocol, the server has no formal output and the output of the client is $\mathcal{Q}(\mathcal{E})$.

An example of range query for encrypted outsourced database is described as follows. $\mathcal{K}$ is an ordered domain of M elements $\{1, ..., N\}$ for $N \in \mathbb{N}$. The family of range queries $Q = \{[i, j]_q\}_{1 \leq i \leq j \leq N}$, where $[i, j]_q(c) = 1$ if $i \leq c \leq j$. Overall, there are $N(N + 1)/2$ (i.e., $\binom{N}{2} + N$) queries in Q. Executing $[i, j]_q$ on a database $\mathcal{E}$ results in all records with search keys in the range $[i, j]$, that is, $[i, j]_q(\mathcal{E}) = \{R_i : i \leq k_i \leq j\}$. For instance, for an employee relation with age column, $\mathcal{K}$ can be essentially viewed as the indexing information of range queries for age. In other words, these keys can be viewed as encrypted indexes for the contents of age, and supports comparisons over encrypted values.

Throughout this paper, for $L, a, b \in \mathbb{N}$, let $[L]$ denote the set $\{1, 2, ..., L\}$, and $[a, b]$ denote the set $[a, a + 1, ..., b]$. This work applies to the general setting of range query scheme as follows. There are two entities: a server and a client. The server stores an encrypted (outsourced) database EDB that a client can query. Let $\{1, 2, ..., N\}$ be the set of possible range query values, that is, N is the maximum value in the range. We denote EDB as a sequence of records corresponding to values in $[1, N]$, and N as the *domain size* of EDB. For an EDB, if there exists at least one record corresponding to each value in $[1, N]$, we say that the EDB is *dense*; otherwise, we say that the EDB is *non-dense*. Given N, there are $N(N+1)/2$ possible range queries. Let $[i, j]_q$ denote a range query issued by a client, where i, j are both integers. Upon receiving a query $[i, j]_q$ from a client, the server returns all records that correspond to the values in $[i, j]$. We assume no information is revealed from the range queries and the corresponding responses between the client and the server, except for the number of records in the response and the access location in the server. The number of records in a response is also called *communication volume* in [23]. In this work, we define the *volume* of a range query $[i, j]_q$ to be the number of records corresponding to the values in $[i, j]$, denoted by $vol([i, j]_q)$. We also define the *count* of a value i in $[N]$ to be the number of records corresponding to value i, i.e., $vol([i, i]_q)$.

2.2 Access location leakage

Access location leakage is a weaker notion than access pattern leakage. As defined in [23], [24], access pattern refers to the information on which records are returned corresponding to a range query. In contrast, access location refers to the attacker learning only the segments of storage when retrieving the group of records corresponding to a range query[2]. For example, in order to rule out access pattern leakage, the client may package the records for each distinct range query as a whole and upload them to the server. However, two identical range queries may still prompt the server to access the same segments of its storage. In a similar manner, for different range queries, the server will access different segments of the storage. Fig. 1 illustrates the difference between the two leakages, in which for two distinct range queries, access location leakage reveals two distinct segments of storage but not the individual records (i.e., Rs in Fig. 1). As an example of the difference between these two leakages, if only access location leakage is available instead of access pattern leakage, the reconstruction attack proposed in [24] may no longer work. To the best of our knowledge, most of known efficient searchable encryption schemes supporting range queries leak access location. In

2. For an adversarial server, it can store the (encrypted) records of a client in a way it preferred such that it can distinguish two different queries by access location leakage. For a scenario where the server honestly stores all the records of a client in a page, if the number of returned tuples for each query is different, the server can still distinguish two distinct queries.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

this work, we employ access location leakage to distinguish between different range queries.

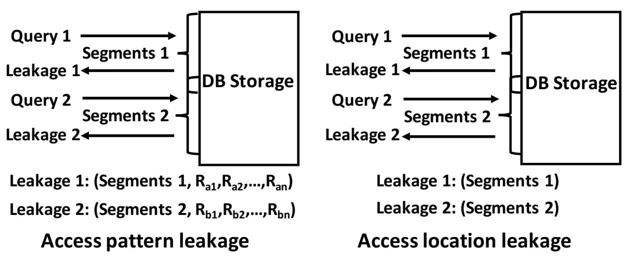


Fig. 1 Comparison between access pattern leakage and access location leakage.

2.3 Adversarial model

An attacker can be the server, or an entity who is able to compromise the server for a period of time. The attacker is *passive* in the sense that it attempts to learn more information than is allowed by examining the information it can observe, but not inject new records as the *active* attacker in [39]. We assume the attacker does not know the underlying range of any issued query. Similar to [15], the attacker learns *how many* records are returned in response to a query. We note that this assumption is reasonable [9], [33]. For the case where the attacker is the server, it can obtain the above information naturally [15]. For the case where the attacker is the one located on the network between the server and the client, as noted in [15], [23], the amount of data being transmitted or the directionality in typical secure communication protocols like TLS are not hidden [9], [33]. For example, the plaintext lengths in ciphertexts are directly leaked in modern TLS cipher suites like those based on AES-GCM [15]. In addition, the attacker can distinguish different query requests via access location leakage (as we assume that it can identify whether a range query is repeated via access location). If the attacker is the server, naturally it has this knowledge. If an attacker is one who compromises the server, it would be able to access this information for the period that the server is compromised. In our attacks, the attacker does not need any knowledge of the database distribution or the query distribution. Since our attacks require every range query to be issued at least once, different query distributions affect the performance of our attacks. This will be discussed during the analysis of our attacks. In addition, different from [15], the attacker in this work does not need to know the total number of records and the domain size (as prior knowledge) [3]. We also do not deal with the query workload attack shown in [14], [25], which allows an attacker to distinguish different query ranges.

3 Passive update recovery attack with one database update

In this section, we present our first attack, namely a practical passive update recovery attack against encrypted database supporting range queries.

3. If the attacker knows the domain size, it will be easier for the attacker to launch the attack since it can be used as prior knowledge directly, and there is no need to recover the domain size N during the second step of the attacks (in Section 3 and Section 4).

Setting of Attack. There is a new record added into the database, and the attacker learns this information. Before and after the addition of the new record, the attacker could collect a set of volumes that covers each possible range query (at least) once, respectively. The database is dense. Since the database is dense, the inserted record is for a value that already exists in the encrypted dataset.
Knowledge of Attacker. Let **T** be the time of adding a new record into the database. The knowledge of the attacker includes: (1) the information that a new record is added into the database at time **T**; (2) the volume set that covers each possible range query once before and after **T**, respectively.
Goal of Attack. There are two goals of our attack. The main goal is to recover the value of newly added record. The second is *database reconstruction*, i.e., to recover the count of each value in $[1, N]$ (before the update of the database).

3.1 Problem Formalization

To better illustrate our attack, we first formalize the underlying problem of our attack setting as follows.
Problem 1. *Alice chooses a random integer N, where $N > 1$. For $i \in [N]$, Alice chooses a random integer x_i and sets $x_i = vol([i,i]_q)$, where $x_i > 0$. Alice computes $vol([i,j]_q) = x_i + x_{i+1} + ... + x_j = A_{[i,j]}$ for $i, j \in [N]$ s.t. $i \leq j$, and sets **A**=$\{A_{[i,j]}\}_{i,j\in[N]\ s.t.\ i\leq j}$. Alice then chooses a random $r \in [N]$ and sets $x_r = x_r + 1$ (i.e., increase x_r by 1). After that, Alice computes $vol([i,j]_q) = x_i + x_{i+1} + ... + x_j = B_{[i,j]}$ for $i, j \in [N]$ s.t. $i \leq j$, and sets **B**=$\{B_{[i,j]}\}_{i,j\in[N]\ s.t.\ i\leq j}$. Finally, Alice gives (**A**,**B**) to Bob. Bob attempts to obtain (1) the value of r; (2) the value of x_i for $i \in [N]$.*
Example 1. Fig. 2 gives an example of our attack setting. As shown in the figure, in the database EDB with domain size $N = 6$, the database counts corresponding to each value in $[N]$ are $\{2, 2, 15, 5, 3, 3\}$. There are a total of 21 distinct range queries. The knowledge of an attacker (for "Case 1 (Step 3)") includes: (1) the volumes of all 21 range queries before and after time **T**, i.e., Knowledge III and Knowledge VI-1 in Fig. 2; (2) a new record is added into EDB at time **T**, i.e., Knowledge IV-1 in Fig. 2. The attacker aims (1) to recover the value of the added record, i.e., Knowledge V-1 in Fig. 2; (2) to recover the exact count of each value in $[1, 6]$, i.e., Knowledge II in Fig. 2.

3.2 Definition and Observation

Full-cover volume set. For a volume set that covers each possible range once, we call it *full-cover volume set*. Since the attacker can identify whether a query is repeated if there is no database update happens, we can obtain two full-cover volume sets. By **A**, **B** we denote the full-cover volume sets before and after the database update.

- **Observation 3.1:** Since there is only one record being added, the volumes in **A** will only increase by 1 if change.
- **Observation 3.2:** Since the database is dense, given **A** and **B**, the following equation holds: $|\mathbf{A}| = |\mathbf{B}| = N(N+1)/2$.

Update-volume set and update-volume count. For our setting of attack, after the database updates, some of the volumes will change. Let **U** be the set containing all volumes that will change when the database update occurs. **U** is called the *update-volume set*, and the size of **U** is called

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

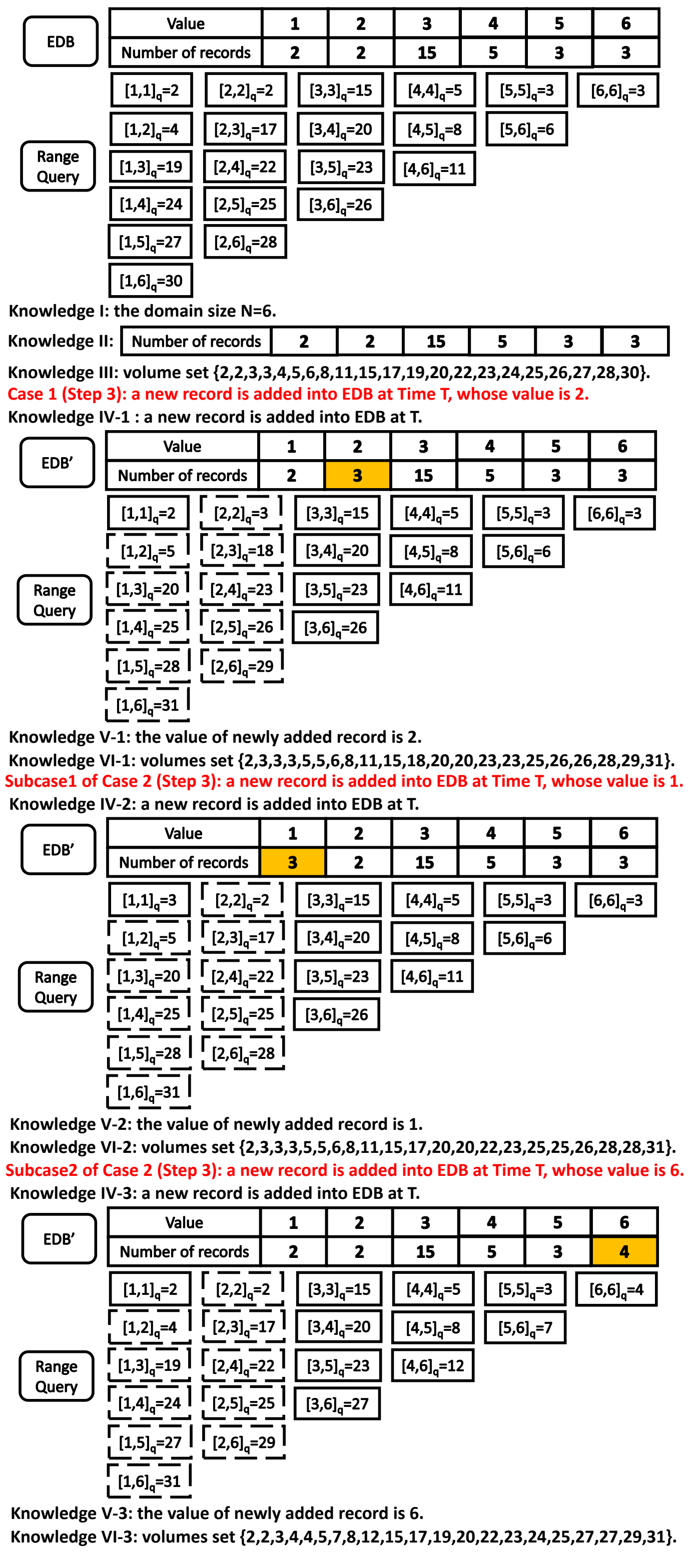

EDB
Value 1 2 3 4 5 6
Number of records 2 2 15 5 3 3
Range Query
[1,1]q=2 [2,2]q=2 [3,3]q=15 [4,4]q=5 [5,5]q=3 [6,6]q=3
[1,2]q=4 [2,3]q=17 [3,4]q=20 [4,5]q=8 [5,6]q=6
[1,3]q=19 [2,4]q=22 [3,5]q=23 [4,6]q=11
[1,4]q=24 [2,5]q=25 [3,6]q=26
[1,5]q=27 [2,6]q=28
[1,6]q=30
Knowledge I: the domain size N=6.
Knowledge II: Number of records 2 2 15 5 3 3
Knowledge III: volume set {2,2,3,3,4,5,6,8,11,15,17,19,20,22,23,24,25,26,27,28,30}.
Case 1 (Step 3): a new record is added into EDB at Time T, whose value is 2.
Knowledge IV-1 : a new record is added into EDB at T.
EDB'
Value 1 2 3 4 5 6
Number of records 2 3 15 5 3 3
Range Query
[1,1]q=2 [2,2]q=3 [3,3]q=15 [4,4]q=5 [5,5]q=3 [6,6]q=3
[1,2]q=5 [2,3]q=18 [3,4]q=20 [4,5]q=8 [5,6]q=6
[1,3]q=20 [2,4]q=23 [3,5]q=23 [4,6]q=11
[1,4]q=25 [2,5]q=26 [3,6]q=26
[1,5]q=28 [2,6]q=29
[1,6]q=31
Knowledge V-1: the value of newly added record is 2.
Knowledge VI-1: volumes set {2,3,3,3,5,5,6,8,11,15,18,20,20,23,23,25,26,26,28,29,31}.
Subcase1 of Case 2 (Step 3): a new record is added into EDB at Time T, whose value is 1.
Knowledge IV-2: a new record is added into EDB at T.
EDB'
Value 1 2 3 4 5 6
Number of records 3 2 15 5 3 3
Range Query
[1,1]q=3 [2,2]q=2 [3,3]q=15 [4,4]q=5 [5,5]q=3 [6,6]q=3
[1,2]q=5 [2,3]q=17 [3,4]q=20 [4,5]q=8 [5,6]q=6
[1,3]q=20 [2,4]q=22 [3,5]q=23 [4,6]q=11
[1,4]q=25 [2,5]q=25 [3,6]q=26
[1,5]q=28 [2,6]q=28
[1,6]q=31
Knowledge V-2: the value of newly added record is 1.
Knowledge VI-2: volumes set {2,3,3,3,5,5,6,8,11,15,17,20,20,22,23,25,25,26,28,28,31}.
Subcase2 of Case 2 (Step 3): a new record is added into EDB at Time T, whose value is 6.
Knowledge IV-3: a new record is added into EDB at T.
EDB'
Value 1 2 3 4 5 6
Number of records 2 2 15 5 3 4
Range Query
[1,1]q=2 [2,2]q=2 [3,3]q=15 [4,4]q=5 [5,5]q=3 [6,6]q=4
[1,2]q=4 [2,3]q=17 [3,4]q=20 [4,5]q=8 [5,6]q=7
[1,3]q=19 [2,4]q=22 [3,5]q=23 [4,6]q=12
[1,4]q=24 [2,5]q=25 [3,6]q=27
[1,5]q=27 [2,6]q=29
[1,6]q=31
Knowledge V-3: the value of newly added record is 6.
Knowledge VI-3: volumes set {2,2,3,4,4,5,7,8,12,15,17,19,20,22,23,25,27,27,29,31}.

Fig. 2 Example of attack setting for passive update recovery attack with one database update.

the *update-volume count*. Given the update-volume count, denoted by C, we have the following observations:

- **Observation 3.3:** Let the value of newly added record be r, then the following equation holds: $C = r(N + 1 - r)$.
- **Observation 3.4:** The smallest element in **U** is $vol([r, r]_q)$.

Elementary and complemented elementary range queries. Similar to [15], we call the range queries $[1, 1]_q$, $[1, 2]_q$, ..., $[1, N]_q$ the *elementary range queries*. We further denote the range queries $[1, N]_q$, $[2, N]_q$, ..., $[N, N]_q$ the *complemented elementary range queries*. We have the following three observations:

- **Observation 3.5:** If the update-volume count equals the domain size (i.e., $C = N$), then the value of newly added record is 1 or N (from **Observation 3.3**), and the update-volume set must be the volume set of elementary range queries or complemented elementary range queries.
- **Observation 3.6:** If one knows the volumes of all elementary range queries, then the count of value t (for $t \in [2, N]$) is the difference between $vol([1, t-1]_q)$ and $vol([1, t]_q)$.
- **Observation 3.7:** If one knows the volumes of all complemented elementary range queries, then the count of value t (for $t \in [N - 1]$) is the difference between $vol([t, N]_q)$ and $vol([t + 1, N]_q)$.

3.3 Attack Overview

Our attack consists of three steps. In **Step 1**, the aim is to obtain the update-volume count. The task of **Step 2** is to compute the domain size N and the value of newly added record. The aim of **Step 3** is to recover the count of value in $[N]$. In particular, if the update-volume count equals N, **Step 3** fully reconstructs the database counts, i.e., it recovers the exact count of each value in $[N]$.

The first procedure in **Step 1** is to collect the full-cover volume set before and after the database update, denoted by **A** and **B**, respectively. Taking **Observation 3.1** into account, we adopts the following approach to obtain the update-volume count: first sort **A** and **B** in ascending order to obtain two new sets $\mathbf{A}'=\{a_1, a_2, ..., a_n\}$ and $\mathbf{B}'=\{b_1, b_2, ..., b_n\}$ respectively, where $n = |\mathbf{A}| = |\mathbf{B}|$. Then count how many a_i satisfying $b_i - a_i = 1$ for $i \in [n]$. The number of such a_i is the update-volume count.

The second step stems from **Observations 3.2, 3.3**. In particular, from **Observation 3.2**, we can recover the domain size N. With N and the update-volume count obtained from **Step 1**, we can recover the value of the added record using **Observation 3.3**.

The third step stems from **Observations 3.4, 3.5, 3.6, 3.7**. For the case where C is not equal to N, from **Observation 3.4**, we can recover the volume of $[r, r]_q$. While for the case where C equals N, we can recover the exact count of each value in $[N]$. Specifically, based on **Observation 3.5**, we first obtain the volume set of elementary range queries or complemented elementary range queries. With the obtained volume set, we can recover the whole database counts using **Observation 3.6** and **Observation 3.7**.

3.4 Description of Attack

We now present the description of our attack.

- **Step 1: Obtain the update-volume count** C**.** First initialize an empty set **S** and collect all volumes. Let **A**, **B** be the full-cover volume set before and after the update of the database, respectively. After obtaining **A** and **B**, sort **A**, **B** in ascending order to obtain two new sets $\mathbf{A}'=\{a_1, a_2, ..., a_n\}$ and $\mathbf{B}'=\{b_1, b_2, ..., b_n\}$ respectively, where n is the size of **A** (or **B**). For $i \in [n]$, for each a_i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

satisfying $b_i - a_i = 1$, add a_i into **S** in ascending order. Finally, obtain $\mathbf{S} = \{a'_1, a'_2, ..., a'_{|\mathbf{S}|}\}$ and $C = |\mathbf{S}|$.

- **Step 2: Recover the domain size N and the value of newly added record r.** With the knowledge of $n = |\mathbf{A}| = |\mathbf{B}|$, obtain N from equation $n = N(N+1)/2$. Then, with the knowledge of C and N, obtain r from equation $C = r(N+1-r)$.
- **Step 3: Recover the count of value in $[N]$.** There are two cases:
  Case 1: $C \neq N$. Find the smallest value in **S**, denoted by a, obtain $vol([r,r]_q) = a$.
  Case 2: $C = N$. Knowing $C = N$, we have that $r = 1$ or $r = N$ from **Step 2**. Hence, **S** is the volume set of elementary range queries or complemented elementary range queries. For $i \in [|\mathbf{S}|]$, do: if $i = 1$, set $s_1 = a'_1$; otherwise, set $s_i = a'_i - a'_{i-1}$. Finally, we have the following two cases:
  - Case 1: $r = 1$. For $i \in [|\mathbf{S}|]$, set $vol([i,i]_q) = s_i$;
  - Case 2: $r = N$. For $i \in [|\mathbf{S}|]$, set $vol([N+1-i, N+1-i]_q) = s_i$.

Example of attack. In the following, we present an example from Fig. 2 to show the process of the attack in Section 3.

For "Case 1 (Step 3)" in Fig. 2:

- **Step 1.** We obtain $\mathbf{A}'$ and $\mathbf{B}'$, where $\mathbf{A}'$ is Knowledge III in Fig. 2 and $\mathbf{B}'$ is Knowledge VI-1 in Fig. 2. With $\mathbf{A}'$ and $\mathbf{B}'$, we can obtain $\mathbf{S} = \{2, 4, 17, 19, 22, 24, 25, 27, 28, 30\}$ and $C = |\mathbf{S}| = 10$.
- **Step 2.** With the knowledge of $n = |\mathbf{A}'| = |\mathbf{B}'| = 21$ and $21 = N(N+1)/2$ (from **Observation 3.2**), we obtain $N = 6$. Knowing $C = |\mathbf{S}| = 10$ and $10 = r(6+1-r)$ (from **Observation 3.3**), we can obtain two candidates of r: 2 or 5.
- **Step 3.** Since $C = 10 \neq N = 6$ and 2 is the smallest value in **S**, we have that $vol([2,2]_q) = 2$ or $vol([5,5]_q) = 2$.

For "Subcase 1 of Case 2 (Step 3)" in Fig. 2:

- **Step 1.** We obtain $\mathbf{A}'$ and $\mathbf{B}'$, where $\mathbf{A}'$ is Knowledge III in Fig. 2 and $\mathbf{B}'$ is Knowledge VI-2 in Fig. 2. With $\mathbf{A}'$ and $\mathbf{B}'$, we can obtain $\mathbf{S} = \{2, 4, 19, 24, 27, 30\}$ and $C = |\mathbf{S}| = 6$.
- **Step 2.** With the knowledge of $n = |\mathbf{A}'| = |\mathbf{B}'| = 21$ and $21 = N(N+1)/2$ (from **Observation 3.2**), we obtain $N = 6$. Knowing $C = |\mathbf{S}| = 6$ and $6 = r(6+1-r)$ (from **Observation 3.3**), we can obtain two candidates of r: 1 or 6.
- **Step 3.** Since $C = N = 6$. For $i \in [6]$, do: if $i = 1$, set $s_1 = 2$; otherwise, set $s_2 = a'_2 - a'_1 = 2$, $s_3 = a'_3 - a'_2 = 15$, $s_4 = a'_4 - a'_3 = 5$, $s_5 = a'_5 - a'_4 = 3$, and $s_6 = a'_6 - a'_5 = 3$. We have the following two cases:
  - Case 1: $r = 1$. For $i \in [6]$, set $vol([1,1]_q) = s_1 = 2$, $vol([2,2]_q) = s_2 = 2$, $vol([3,3]_q) = s_3 = 15$, $vol([4,4]_q) = s_4 = 5$, $vol([5,5]_q) = s_5 = 3$, and $vol([6,6]_q) = s_6 = 3$.
  - Case 2: $r = 6$. For $i \in [6]$, set $vol([6,6]_q) = s_1 = 2$, $vol([5,5]_q) = s_2 = 2$, $vol([4,4]_q) = s_3 = 15$, $vol([3,3]_q) = s_4 = 5$, $vol([2,2]_q) = s_5 = 3$, and $vol([1,1]_q) = s_6 = 3$.

For "Subcase 2 of Case 2 (Step 3)" in Fig. 2:

- **Step 1.** We obtain $\mathbf{A}'$ and $\mathbf{B}'$, where $\mathbf{A}'$ is Knowledge III in Fig. 2 and $\mathbf{B}'$ is Knowledge VI-3 in Fig. 2. With $\mathbf{A}'$ and $\mathbf{B}'$, we can obtain $\mathbf{S} = \{3, 6, 11, 26, 28, 30\}$ and $C = |\mathbf{S}| = 6$.
- **Step 2.** With the knowledge of $n = |\mathbf{A}'| = |\mathbf{B}'| = 21$ and $21 = N(N+1)/2$ (from **Observation 3.2**), we obtain $N = 6$. Knowing $C = |\mathbf{S}| = 6$ and $6 = r(6+1-r)$ (from **Observation 3.3**), we can obtain two candidates of r: 1 or 6.
- **Step 3.** Since $C = N = 6$. For $i \in [6]$, do: if $i = 1$, set $s_1 = 3$; otherwise, set $s_2 = a'_2 - a'_1 = 3$, $s_3 = a'_3 - a'_2 = 5$, $s_4 = a'_4 - a'_3 = 15$, $s_5 = a'_5 - a'_4 = 2$, and $s_6 = a'_6 - a'_5 = 2$. We have the following two cases:
  - Case 1: $r = 1$. For $i \in [6]$, set $vol([1,1]_q) = s_1 = 3$, $vol([2,2]_q) = s_2 = 3$, $vol([3,3]_q) = s_3 = 5$, $vol([4,4]_q) = s_4 = 15$, $vol([5,5]_q) = s_5 = 2$, and $vol([6,6]_q) = s_6 = 2$.
  - Case 2: $r = 6$. For $i \in [6]$, set $vol([6,6]_q) = s_1 = 3$, $vol([5,5]_q) = s_2 = 3$, $vol([4,4]_q) = s_3 = 5$, $vol([3,3]_q) = s_4 = 15$, $vol([2,2]_q) = s_5 = 2$, and $vol([1,1]_q) = s_6 = 2$.

3.5 Explanation and Analysis

Explanation and analysis of Step 1. **Step 1** consists of three procedures, we present the analysis of each procedure as follows.

- The first procedure is to collect all volumes. Since the attacker could identify whether the same query has been issued via the access of storage and the database is dense, the attacker needs to observe $N(N+1)/2$ distinct range queries (and obtain the corresponding volumes). Similar to [15], if we assume the query distribution is uniform, the volume collection procedure is the classic coupon collector's problem. In our setting, there are $\mathcal{O}(N^2)$ possible range queries, coupon collection implies that $\mathcal{O}(N^2 \log N)$ range queries suffice. Otherwise, if the query distribution is non-uniform, as noted in [15], $\mathcal{O}(\alpha^{-1} N^2 \log N)$ range queries suffice if the least likely range has probability $\frac{\alpha}{N(N+1)/2}$.
- The second procedure is to sort **A** and **B** in ascending order. The complexity is $\mathcal{O}(n \log n)$ if Heap Sort or Merge Sort is used.
- The third procedure is to extract each a_i satisfying $b_i - a_i = 1$ from **A**. The complexity is $\mathcal{O}(n)$. Based on **Observation 3.1**, this procedure can help us identify the set of volume that will change when the database update occurs.

Explanation and analysis of Step 2. There are two procedures in **Step 2**, we now present the analysis of each procedure as follows.

- The first procedure is to recover N. Since the attacker can identify whether the same range query is repeated and the database is dense, it can obtain exactly $N(N+1)/2$ volumes corresponding to distinct range queries. Based on **Observation 3.2**, this equals the size of **A** (or **B**). Hence, it is easy for the attacker to compute N. The complexity is to solve a quadratic equation.
- The second procedure is to obtain r. From **Observation 3.3**, we have $C = r(N+1-r)$. Since the attacker knows C (from **Step 1**) and N (from the first procedure shown above), it is easy to compute the value of r. The complexity is to solve a quadratic equation. Note that the value of newly added record can only be recovered up

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

to reflection, that is, for any value r, the recovered value could be for r or $N+1-r$.

Explanation and analysis of Step 3. This step consists of two cases. We give the analysis of each case as follows.

- **Case 1:** In this case, based on **Observation 3.4**, the volume of $[r,r]_q$ is the smallest one in **S**.
- **Case 2:** Since $C = N$, from **Step 2**, we have that $r = 1$ or $r = N$. Based on **Observation 3.5**, we conclude that **S** is the volume set of elementary or complemented elementary range queries. If $r = 1$, from **Observation 3.6**, we have that $a'_i - a'_{i-1}$ must be the volume of $[i,i]$ for $1 < i \leq N$. Likewise, if $r = N$, from **Observation 3.7**, we have that $a'_i - a'_{i-1}$ must be the volume of $[N+1-i, N+1-i]$ for $1 \leq i < N$. The complexity is $\mathcal{O}(N)$.

Explanations of observations. **Observations 3.1, 3.2, 3.4, 3.6, 3.7** are straightforward, in the following we mainly present the explanations of **Observation 3.3** and **Observation 3.5**.

- **Explanation of Observation 3.3.** Note that the update-volume count reflects the number of range queries containing the value of newly added record r. Hence, if we can calculate the number of range queries containing r, then we can obtain the update-volume count. For a range query $[a,b]$, if it contains r, then one of the following cases must be satisfied: (1) Case 1: $a = r$ and $a \leq b \leq N$; (2) Case 2: $1 \leq a < r$ and $r \leq b \leq N$. In Case 1, there are $N+1-r$ possibilities. In Case 2, there are $(r-1)(N+1-r)$ possibilities. Combining Case 1 and Case 2, the total possibilities are $r(N+1-r)$. Hence, there are $r(N+1-r)$ range queries containing r.
- **Explanation of Observation 3.5.** For all range queries, only $[1,1]_q$, $[1,2]_q$,..., $[1,N]_q$ contain value 1, which are the elementary range queries. Hence, if the value of newly added record r is 1, then the update-volume set must be the volume set of elementary range queries. Likewise, for all range queries, only $[1,N]_q$, $[2,N]_q$,..., $[N,N]_q$ contain value N, which are the complemented elementary range queries. Hence, if the value of newly added record r is N, then the update-volume set must be the volume set of complemented elementary range queries.

4 Passive update recovery attack with two updates

The attack in Section 3 mainly deals with the case of only one database update. In this section, we consider a setting where there are two database updates.

Setting of Attack. The setting is the same with that of the attack in Section 3, except that there is one more database update in the sense that another record with a different value is added[4]. Since the database is dense, the two inserted records are for two values that already exist in the encrypted dataset.

Knowledge of Attacker. By $\mathbf{T}_1$, $\mathbf{T}_2$ we denote the times of adding the first record and the second record into the database, respectively. The knowledge of the attacker includes: (1) the information that two different records are added into the database at $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively; (2) the volume set that covers each possible range query once before $\mathbf{T}_1$, after $\mathbf{T}_1$ but before $\mathbf{T}_2$, and after $\mathbf{T}_2$, respectively.

Goal of Attack. There are two goals of the attack. The main goal is to recover the values of newly added records. The second is to recover the count of each value in $[1,N]$ before the first update.

4. We note that the values of the two records are different since if they are identical, the setting is the same as Section 3.

4.1 Problem Formalization

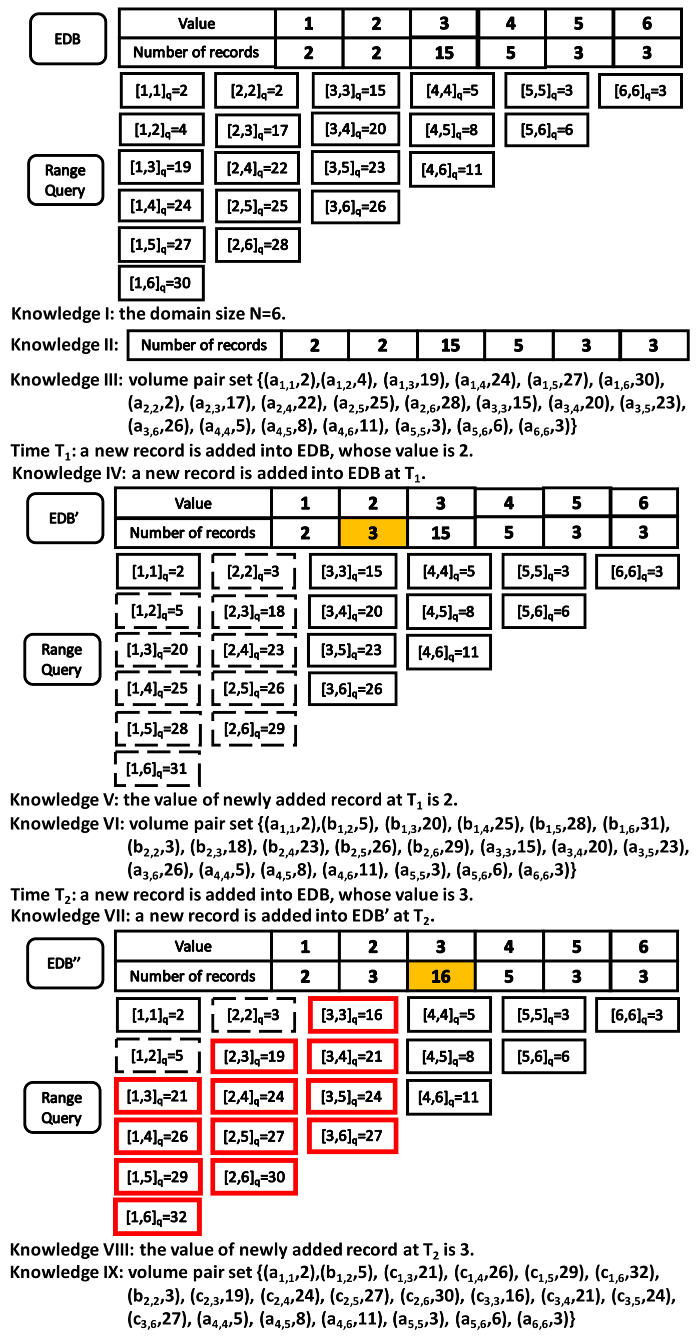
We formalize the underlying problem of our attack setting in this section as follows.

Problem 2. *Alice chooses a random integer N, where $N > 1$. For $i \in [N]$, Alice chooses a random integer x_i (where $x_i > 0$), $N(N+1)/2$ distinct integers (each chosen integer is denoted by $a_{i,j}$ for $i,j \in [N]$ s.t. $i \leq j$), and sets $x_i = vol([i,i]_q)$. Alice then computes $vol([i,j]_q) = x_i + x_{i+1} + ... + x_j = A_{[i,j]}$ for $i,j \in [N]$ s.t. $i \leq j$, and sets **A**=$\{(a_{i,j}, A_{[i,j]})\}_{i,j\in[N]\ s.t.\ i\leq j}$. At time $\mathbf{T}_1$, Alice chooses a random $r_1 \in [N]$ and sets $x_{r_1} = x_{r_1}+1$ (i.e., increase x_{r_1} by 1). In addition, for $i,j \in [N]$ s.t. $i \leq j$, she sets $b_{i,j} = a_{i,j}$ if $r_1 \notin [i,j]$ and chooses a new distinct integer $b_{i,j}$ if $r_1 \in [i,j]$. After that, she computes $vol([i,j]_q) = x_i + x_{i+1} + ... + x_j = B_{[i,j]}$ for $i,j \in [N]$ s.t. $i \leq j$, and sets **B**=$\{(b_{i,j}, B_{[i,j]})\}_{i,j\in[N]\ s.t.\ i\leq j}$. At time $\mathbf{T}_2$, Alice chooses a random $r_2 \in [N]$ and sets $x_{r_2} = x_{r_2}+1$ (i.e., increase x_{r_2} by 1). In addition, for $i,j \in [N]$ s.t. $i \leq j$, she sets $c_{i,j} = b_{i,j}$ if $r_2 \notin [i,j]$ and chooses a new distinct integer $c_{i,j}$ if $r_2 \in [i,j]$. After that, she computes $vol([i,j]_q) = x_i + x_{i+1} + ... + x_j = C_{[i,j]}$ for $i,j \in [N]$ s.t. $i \leq j$, and sets **C**=$\{(c_{i,j}, C_{[i,j]})\}_{i,j\in[N]\ s.t.\ i\leq j}$. Finally, Alice gives (**A**,**B**,**C**) to Bob. Bob attempts to obtain (1) the values of r_1 and r_2; (2) the value of x_i for $i \in [N]$.*

Example 2. Fig. 3 gives an example of the attack setting in this section. The database EDB is the same with the database in Fig. 2. In this example, the knowledge of an attacker includes: (1) the volume pairs of all 21 range queries before time $\mathbf{T}_1$, after time $\mathbf{T}_1$ but before time $\mathbf{T}_2$, and after time $\mathbf{T}_2$, respectively, i.e., Knowledge III, Knowledge VI and Knowledge IX in Fig. 3; (2) two new records are added into EDB at time $\mathbf{T}_1$ and time $\mathbf{T}_2$ separately, i.e., Knowledge IV and Knowledge VII in Fig. 3. The target of the attacker include (1) to recover the value of each added record, i.e., Knowledge V and VIII in Fig. 3; (2) to recover the exact count of each value in $[1,6]$, i.e., Knowledge II in Fig. 3.

4.2 Notation

Let **X**, **Y** be pair sets consists of pairs $(x_{1,i}, x_{2,i})_{i\in[n]}$, $(y_{1,i}, y_{2,i})_{i\in[n]}$ respectively, whereby if $x_{1,i} = y_{1,i}$ then $x_{2,i} = y_{2,i}$. By $\mathbf{X} \cap \mathbf{Y}$ we denote the intersection of **X** and **Y**, i.e., $\mathbf{X} \cap \mathbf{Y} = \{(x_{1,i}, x_{2,i})\ or\ (y_{1,i},\ y_{2,i}) | x_{1,i} = y_{1,i}\ \&\ x_{2,i} = y_{2,i}\}_{i\in[n]}$. By $\mathbf{X} - \mathbf{Y}$ we denote the subset of **X** that is different from **Y** according to the first entry of each pair, i.e., $\mathbf{X} - \mathbf{Y} = \{(x_{1,i}, x_{2,i}) | x_{1,i} \neq y_{1,j}\}_{i,j\in[n]}$. Let $\mathbf{X}^2$ be the set containing the second entry of each pair in **X**, i.e., $\mathbf{X}^2 = \{x_{2,i}\}_{i\in[n]}$. By $\mathbf{X}^{2(-1)}$ we denote the set containing the result of each element in $\mathbf{X}^2$ minus by 1, i.e., $\mathbf{X}^{2(-1)} = \{x_{2,i} - 1 | x_{2,i} \in \mathbf{X}^2\}_{i\in[n]}$. Similarly, by $\mathbf{X}^{2(+1)}$ we denote the set containing the result of each element in $\mathbf{X}^2$ increase by 1, i.e., $\mathbf{X}^{2(+1)} = \{x_{2,i} + 1 | x_{2,i} \in \mathbf{X}^2\}_{i\in[n]}$. By $\mathbf{X}^2 - \mathbf{Y}^2$ we denote the set containing all elements in $\mathbf{X}^2$ but not in $\mathbf{Y}^2$.


EDB
Value 1 2 3 4 5 6
Number of records 2 2 15 5 3 3
Range Query
$[1,1]_q$=2 $[2,2]_q$=2 $[3,3]_q$=15 $[4,4]_q$=5 $[5,5]_q$=3 $[6,6]_q$=3
$[1,2]_q$=4 $[2,3]_q$=17 $[3,4]_q$=20 $[4,5]_q$=8 $[5,6]_q$=6
$[1,3]_q$=19 $[2,4]_q$=22 $[3,5]_q$=23 $[4,6]_q$=11
$[1,4]_q$=24 $[2,5]_q$=25 $[3,6]_q$=26
$[1,5]_q$=27 $[2,6]_q$=28
$[1,6]_q$=30
EDB'
Value 1 2 3 4 5 6
Number of records 2 3 15 5 3 3
$[1,1]_q$=2 $[2,2]_q$=3 $[3,3]_q$=15 $[4,4]_q$=5 $[5,5]_q$=3 $[6,6]_q$=3
$[1,2]_q$=5 $[2,3]_q$=18 $[3,4]_q$=20 $[4,5]_q$=8 $[5,6]_q$=6
$[1,3]_q$=20 $[2,4]_q$=23 $[3,5]_q$=23 $[4,6]_q$=11
$[1,4]_q$=25 $[2,5]_q$=26 $[3,6]_q$=26
$[1,5]_q$=28 $[2,6]_q$=29
$[1,6]_q$=31
EDB''
Value 1 2 3 4 5 6
Number of records 2 3 16 5 3 3
$[1,1]_q$=2 $[2,2]_q$=3 $[3,3]_q$=16 $[4,4]_q$=5 $[5,5]_q$=3 $[6,6]_q$=3
$[1,2]_q$=5 $[2,3]_q$=19 $[3,4]_q$=21 $[4,5]_q$=8 $[5,6]_q$=6
$[1,3]_q$=21 $[2,4]_q$=24 $[3,5]_q$=24 $[4,6]_q$=11
$[1,4]_q$=26 $[2,5]_q$=27 $[3,6]_q$=27
$[1,5]_q$=29 $[2,6]_q$=30
$[1,6]_q$=32

Knowledge I: the domain size N=6.

Knowledge II: Number of records 2 2 15 5 3 3

Knowledge III: volume pair set {($a_{1,1}$,2),($a_{1,2}$,4), ($a_{1,3}$,19), ($a_{1,4}$,24), ($a_{1,5}$,27), ($a_{1,6}$,30), ($a_{2,2}$,2), ($a_{2,3}$,17), ($a_{2,4}$,22), ($a_{2,5}$,25), ($a_{2,6}$,28), ($a_{3,3}$,15), ($a_{3,4}$,20), ($a_{3,5}$,23), ($a_{3,6}$,26), ($a_{4,4}$,5), ($a_{4,5}$,8), ($a_{4,6}$,11), ($a_{5,5}$,3), ($a_{5,6}$,6), ($a_{6,6}$,3)}

Time T_1: a new record is added into EDB, whose value is 2.

Knowledge IV: a new record is added into EDB at T_1.

Knowledge V: the value of newly added record at T_1 is 2.

Knowledge VI: volume pair set {($a_{1,1}$,2),($b_{1,2}$,5), ($b_{1,3}$,20), ($b_{1,4}$,25), ($b_{1,5}$,28), ($b_{1,6}$,31), ($b_{2,2}$,3), ($b_{2,3}$,18), ($b_{2,4}$,23), ($b_{2,5}$,26), ($b_{2,6}$,29), ($a_{3,3}$,15), ($a_{3,4}$,20), ($a_{3,5}$,23), ($a_{3,6}$,26), ($a_{4,4}$,5), ($a_{4,5}$,8), ($a_{4,6}$,11), ($a_{5,5}$,3), ($a_{5,6}$,6), ($a_{6,6}$,3)}

Time T_2: a new record is added into EDB, whose value is 3.

Knowledge VII: a new record is added into EDB' at T_2.

Knowledge VIII: the value of newly added record at T_2 is 3.

Knowledge IX: volume pair set {($a_{1,1}$,2),($b_{1,2}$,5), ($c_{1,3}$,21), ($c_{1,4}$,26), ($c_{1,5}$,29), ($c_{1,6}$,32), ($b_{2,2}$,3), ($c_{2,3}$,19), ($c_{2,4}$,24), ($c_{2,5}$,27), ($c_{2,6}$,30), ($c_{3,3}$,16), ($c_{3,4}$,21), ($c_{3,5}$,24), ($c_{3,6}$,27), ($a_{4,4}$,5), ($a_{4,5}$,8), ($a_{4,6}$,11), ($a_{5,5}$,3), ($a_{5,6}$,6), ($a_{6,6}$,3)}

Fig. 3 Example of attack setting for passive update recovery attack with two database updates.

4.3 Definition and Observation

Full-cover access-volume pair set. Recall the attacker can distinguish different range queries by observing the access of its storage. Hence, given a range query, the attacker knows the volume as well as its access of storage. We define the pair (access location,volume) as a *access-volume pair*. For a set consisting of access-volume pairs that covers each possible range query once, we call it *full-cover access-volume pair set*. Intuitively, we can obtain three full-cover access-volume pair sets. By **A**, **B**, **C** we denote the full-cover access-volume pair set before the first database update, after the first database update but before the second database update, and after the second database update, respectively.

- **Observation 4.1:** Similar to **Observation 3.2**, since the database is dense, given **A**, **B**, **C**, the following equation holds: $|\mathbf{A}| = |\mathbf{B}| = |\mathbf{C}| = N(N+1)/2$.

Update-access-volume set, update-access-volume count and update-intersection count. For our setting, some access-volume pairs will change when the database update occurs. For a set **U** containing all access-volume pairs whose values *will change* once the database update happens, we call it the *Type-I update-access-volume set*. For a set **U** containing all access-volume pairs whose values *have been changed* after the database update, we call it the *Type-II update-access-volume set*. We call the size of Type-I or Type-II update-access-volume set the *update-access-volume count*. If set $\mathbf{U}_1$ is the Type-II update-access-volume set for the first update and set $\mathbf{U}_2$ is the Type-I update-access-volume set for the second update, we call the intersection of $\mathbf{U}_1$ and $\mathbf{U}_2$ the *update-intersection set* and the size of the update-intersection set the *update-intersection count*.

- **Observation 4.2:** The set containing all the access-volume pairs in **A** that are different from any access-volume pair in **B** is the Type-I update-access-volume set for the first update, and its size is the corresponding update-access-volume count. The set containing all the access-volume pairs in **B** that are different from any access-volume pair in **A** is the Type-II update-access-volume set for the first update, and its size is the corresponding update-access-volume count. Similarly, the set containing all the access-volume pairs in **B** that are different from any access-volume pair in **C** is the Type-I update-access-volume set for the second update, and its size is the corresponding update-access-volume count. The set containing all the access-volume pairs in **C** that are different from any access-volume pair in **B** is the Type-II update-access-volume set for the second update, and its size is the corresponding update-access-volume count.
- **Observation 4.3:** Similar to **Observation 3.3**, let C be the update-access-volume count and the value of newly added record be r, the following equation holds: $C = r(N+1-r)$.
- **Observation 4.4:** Let the value of the first added record be r_1, the value of the second added record be r_2, $\mathbf{U}_{1,1}$ be the Type-I update-access-volume set for the first update, $\mathbf{U}_{1,2}$ be the Type-II update-access-volume set for the first update, $\mathbf{U}_{2,1}$ be the Type-I update-access-volume set for the second update, $\mathbf{U}_{2,2}$ be the Type-II update-access-volume set for the second update. From **Observation 4.3**, the recovered value of newly added record will have two possible values, up to reflection. Without loss of generality, let $\widehat{r}_1$ and $N+1-\widehat{r}_1$ be the two possible values of r_1, $\widehat{r}_2$ and $N+1-\widehat{r}_2$ be the two possible values of r_2, where $\widehat{r}_1 \leq (N+1)/2 \leq N+1-\widehat{r}_1, \widehat{r}_2 \leq (N+1)/2 \leq N+1-\widehat{r}_2$. In addition, Let x_1 be the smallest volume in $\mathbf{U}^2_{1,1}$, x_2 be the smallest volume in $\mathbf{U}^2_{2,1}$. We can obtain two sets $\mathbf{S}_1 = \mathbf{U}_{1,2} - \mathbf{U}_{1,2} \cap \mathbf{U}_{2,1}$ and $\mathbf{S}_2 = \mathbf{U}_{2,1} - \mathbf{U}_{1,2} \cap \mathbf{U}_{2,1}$, and if we sort the elements in $\mathbf{S}_1^{2(-1)}$, $\mathbf{S}_2^2$ in ascending order, we will obtain sets $\{s_{1,1}, s_{1,2}, ..., s_{1,\widehat{s}_1}\}$, $\{s_{2,1}, s_{2,2}, ..., s_{2,\widehat{s}_2}\}$, respectively. We have:
  - If $(\widehat{r}_1, \widehat{r}_2)$ is the result, we have two cases:
    - $|\widehat{r}_1 - \widehat{r}_2| \neq 1$: we have $vol([\widehat{r}_1, \widehat{r}_1]_q) = x_1$, $vol([\widehat{r}_2, \widehat{r}_2]_q) = x_2$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

  - $|\widehat{r}_1 - \widehat{r}_2| = 1$: we have the following two cases:
    (1) $\widehat{r}_2 = \widehat{r}_1 + 1$: we have $\widehat{s}_1 = \widehat{r}_1$, $\widehat{s}_2 = N + 1 - \widehat{r}_2$, $\{vol([\widehat{r}_1 + 1 - i, \widehat{r}_1]) = s_{1,i}\}_{i \in [\widehat{r}_1]}$, $\{vol([\widehat{r}_2, i + \widehat{r}_2 - 1]) = s_{2,i}\}_{i \in [N+1-\widehat{r}_2]}$.
    (2) $\widehat{r}_1 = \widehat{r}_2 + 1$: we have $\widehat{s}_1 = N + 1 - \widehat{r}_1$, $\widehat{s}_2 = \widehat{r}_2$, $\{vol([\widehat{r}_1, i + \widehat{r}_1 - 1]) = s_{1,i}\}_{i \in [N+1-\widehat{r}_1]}$, $\{vol([\widehat{r}_2 + 1 - i, \widehat{r}_2]) = s_{2,i}\}_{i \in [\widehat{r}_2]}$.
- If $(\widehat{r}_1, N + 1 - \widehat{r}_2)$ is the result, we have $vol([\widehat{r}_1, \widehat{r}_1]_q) = x_1$, $vol([N + 1 - \widehat{r}_2, N + 1 - \widehat{r}_2]_q) = x_2$;
- If $(N + 1 - \widehat{r}_1, \widehat{r}_2)$ is the result, we have $vol([N + 1 - \widehat{r}_1, N + 1 - \widehat{r}_1]_q) = x_1$, $vol([\widehat{r}_2, \widehat{r}_2]_q) = x_2$;
- If $(N + 1 - \widehat{r}_1, N + 1 - \widehat{r}_2)$ is the result, we have two cases:
  - $|N+1-\widehat{r}_1-(N+1-\widehat{r}_2)| \neq 1$, we have $vol([N+1-\widehat{r}_1, N+1-\widehat{r}_1]_q) = x_1$, $vol([N+1-\widehat{r}_2, N+1-\widehat{r}_2]_q) = x_2$;
  - $|N + 1 - \widehat{r}_1 - (N + 1 - \widehat{r}_2)| = 1$, we have two cases:
    (1) $N + 1 - \widehat{r}_2 = (N + 1 - \widehat{r}_1) + 1$: we have $\widehat{s}_1 = N+1-\widehat{r}_1$, $\widehat{s}_2 = \widehat{r}_2$, $\{vol([N+2-\widehat{r}_1-i, N+1-\widehat{r}_1]) = s_{1,i}\}_{i \in [N+1-\widehat{r}_1]}$, $\{vol([N + 1 - \widehat{r}_2, i + N - \widehat{r}_2]) = s_{2,i}\}_{i \in [\widehat{r}_2]}$.
    (2) $N+1-\widehat{r}_1 = (N+1-\widehat{r}_2)+1$: we have $\widehat{s}_1 = \widehat{r}_1$, $\widehat{s}_2 = N+1-\widehat{r}_2$, $\{vol([N+1-\widehat{r}_1, i+N-\widehat{r}_1]) = s_{1,i}\}_{i \in [\widehat{r}_1]}$, $\{vol([N+2-\widehat{r}_2-i, N+1-\widehat{r}_2]) = s_{2,i}\}_{i \in [N+1-\widehat{r}_2]}$.

- **Observation 4.5:** Given $\widehat{r}_1$, $N + 1 - \widehat{r}_1$, $\widehat{r}_2$, $N + 1 - \widehat{r}_2$ defined in **Observation 4.4**, we have:
  - If $(\widehat{r}_1, \widehat{r}_2)$ is the result, we have two cases:
    - If $\widehat{r}_1 < \widehat{r}_2$, the update-intersection count is $\widehat{r}_1(N + 1 - \widehat{r}_2)$;
    - If $\widehat{r}_2 < \widehat{r}_1$, the update-intersection count is $\widehat{r}_2(N + 1 - \widehat{r}_1)$.
  - If $(\widehat{r}_1, N + 1 - \widehat{r}_2)$ is the result, the update-intersection count is $\widehat{r}_1 \cdot \widehat{r}_2$;
  - If $(N + 1 - \widehat{r}_1, \widehat{r}_2)$ is the result, the update-intersection count is $\widehat{r}_1 \cdot \widehat{r}_2$;
  - If $(N + 1 - \widehat{r}_1, N + 1 - \widehat{r}_2)$ is the result, we have two cases:
    - If $N+1-\widehat{r}_1 < N+1-\widehat{r}_2$, then the update-intersection count is $\widehat{r}_2(N + 1 - \widehat{r}_1)$;
    - If $N+1-\widehat{r}_2 < N+1-\widehat{r}_1$, then the update-intersection count is $\widehat{r}_1(N + 1 - \widehat{r}_2)$.
- **Observation 4.6:** Given $(r_2 = r_1 + 1, \{vol([i, r_1]_q)\}_{i \in [r_1]}, \{vol([r_2, i]_q)\}_{i \in [r_2, N]})$ or $(r_1 = r_2 + 1, \{vol([r_1, i]_q)\}_{i \in [r_1, N]}, \{vol([i, r_2]_q)\}_{i \in [r_2]})$, we can recover the counts of values in $[N]$ as follows:
  - For $(r_2 = r_1 + 1, \{vol([i, r_1]_q)\}_{i \in [r_1]}, \{vol([r_2, i]_q)\}_{i \in [r_2, N]})$, compute $vol([i, i]_q) = vol([i, r_1]_q) - vol([i + 1, r_1]_q)$ for $i \in [r_1 - 1]$, $vol([r_2 + i, r_2 + i]_q) = vol([r_2, i + r_2]_q) - vol([r_2, i + r_2 - 1]_q)$ for $i \in [N - r_2]$;
  - For $(r_1 = r_2 + 1, \{vol([r_1, i]_q)\}_{i \in [r_1, N]}, \{vol([i, r_2]_q)\}_{i \in [r_2]})$, compute $vol([i, i]_q) = vol([i, r_2]_q) - vol([i + 1, r_2]_q)$ for $i \in [r_2 - 1]$, $vol([r_1 + i, r_1 + i]_q) = vol([r_1, i + r_1]_q) - vol([r_1, i + r_1 - 1]_q)$ for $i \in [N - r_1]$.

4.4 Attack Overview

The attack consists of three steps. In **Step 1**, we aim to obtain the update-access-volume count and the update-intersection count. **Step 2** is to recover the domain size and the values of newly added records. Finally, we recover the count of value in $[N]$ in **Step 3**. In particular, if the difference between the values of the two added records is 1, **Step 3** can recover the count of each value in $[N]$.

The first procedure in **Step 1** is to collect the full-cover access-volume pair set before the first update, denoted by **A**, after the first update but before the second update, denoted by **B**, and after the second update, denoted by **C**. With **A**, **B** and **C**, based on **Observation 4.2**, we further obtain the following sets: (1) the set containing the access-volume pairs that have been changed after the first update (i.e., Type-II update-access-volume set for the first update), denoted by **D**; (2) the set containing the access-volume pairs that will change when the second update occurs (i.e., Type-I update-access-volume set for the second update), denoted by **E**; (3) the set containing the access-volume pairs that have been changed after the first update and will change when the second update occurs, denoted by **F**; (4) the set containing the access-volume pairs that will change when the first update occurs, denoted by **G**. Finally, we can obtain the update-access-volume count for the first update, the update-access-volume count for the second update and the update-intersection count, which is the size of **D**, **E** and **F**, respectively.

Step 2 stems from **Observations 4.1, 4.3, 4.5**. In particular, we can easily recover the domain size N based on **Observation 4.1**. With N and the update-access-volume counts for the first update and the second update obtained from **Step 1**, we can recover four candidate pairs of the values of newly added records using **Observation 4.3**. To check whether the candidate pair is the result, the attack runs a further checking step based on **Observation 4.5**.

Step 3 stems from **Observations 3.5, 4.4**. The main task of this step is to recover the count of value in $[N]$. If **D** or **E** equals N, we can recover the count of each value in $[N]$ as in Section 3. Otherwise, we employ another approach. Specifically, there are two cases. Let r_1, r_2 be the recovered values of newly added records. For the case where $|r_1 - r_2| \neq 1$, the attack can only recover the counts of $[r_1, r_1]_q$ and $[r_2, r_2]_q$. For the case where $|r_1 - r_2| = 1$, the attack can fully reconstruct the counts of the whole database.

4.5 Description of Attack

The attack consists of the following steps:

- **Step 1: Obtain the update-access-volume count and the update-intersection count.** Let **A**, **B**, and **C** be the full-cover access-volume pair sets before $\mathbf{T}_1$, after $\mathbf{T}_1$ but before $\mathbf{T}_2$, and after $\mathbf{T}_2$, respectively. Obtain set $\mathbf{D} = \mathbf{B} - \mathbf{A} \cap \mathbf{B}$, $\mathbf{E} = \mathbf{B} - \mathbf{B} \cap \mathbf{C}$, and $\mathbf{F} = \mathbf{D} \cap \mathbf{E}$. In addition, obtain $\mathbf{G} = \mathbf{A} - \mathbf{A} \cap \mathbf{B}$ for future use (in **Step 3**). Finally, obtain the update-access-volume count for the first update $|\mathbf{D}|$, the update-access-volume count for the second update $|\mathbf{E}|$, and the update-intersection count $|\mathbf{F}|$.
- **Step 2: Recover the domain size N, the value of the first added record r_1 and the value of the second added record r_2.** Let n be the size of **A** (or **B**, or **C**), first obtain N from equation $n = N(N + 1)/2$. With the knowledge of $|\mathbf{D}|$, obtain r_1 from equation $|\mathbf{D}| = r_1(N + 1 - r_1)$, denoted by $\widehat{r}_1$ or $N + 1 - \widehat{r}_1$, where $\widehat{r}_1 \leq (N + 1)/2 \leq N + 1 - \widehat{r}_1$; similarly, with the knowledge of $|\mathbf{E}|$, obtain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

r_2 from equation $|\mathbf{E}| = r_2(N+1-r_2)$, denoted by $\widehat{r}_2$ or $N+1-\widehat{r}_2$, where $\widehat{r}_2 \le (N+1)/2 \le N+1-\widehat{r}_2$. So, we have the following candidate pairs of (r_1, r_2): $(\widehat{r}_1, \widehat{r}_2)$, $(\widehat{r}_1, N+1-\widehat{r}_2)$, $(N+1-\widehat{r}_1, \widehat{r}_2)$, $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$[5]. We further process the above candidate pairs as follows:

- For candidate pair $(\widehat{r}_1, \widehat{r}_2)$,
  * if $\widehat{r}_1 < \widehat{r}_2$, check whether the equation $\widehat{r}_1(N+1-\widehat{r}_2) = |\mathbf{F}|$ holds, if yes, record $(\widehat{r}_1, \widehat{r}_2)$ as the final result;
  * if $\widehat{r}_2 < \widehat{r}_1$, check whether the equation $\widehat{r}_2(N+1-\widehat{r}_1) = |\mathbf{F}|$ holds, if yes, record $(\widehat{r}_1, \widehat{r}_2)$ as the final result.
- For candidate pair $(\widehat{r}_1, N+1-\widehat{r}_2)$, check whether the equation $\widehat{r}_1 \cdot \widehat{r}_2 = |\mathbf{F}|$ holds, if yes, record $(\widehat{r}_1, N+1-\widehat{r}_2)$ as the final result;
- For candidate pair $(N+1-\widehat{r}_1, \widehat{r}_2)$, check whether the equation $\widehat{r}_1 \cdot \widehat{r}_2 = |\mathbf{F}|$ holds, if yes, record $(R+1-\widehat{r}_1, \widehat{r}_2)$ as the final result;
- For candidate pair $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$,
  * if $N+1-\widehat{r}_1 < N+1-\widehat{r}_2$, check whether the equation $(N+1-\widehat{r}_1)\widehat{r}_2 = |\mathbf{F}|$ holds, if yes, record $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$ as the final result;
  * if $N+1-\widehat{r}_2 < N+1-\widehat{r}_1$, check whether the equation $(N+1-\widehat{r}_2)\widehat{r}_1 = |\mathbf{F}|$ holds, if yes, record $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$ as the final result.

- **Step 3: Recover the count of value in $[N]$.** First check whether $|\mathbf{D}|$ or $|\mathbf{E}|$ equals N, if yes, let $\mathbf{S} = \{a_1, a_2, ..., a_{|\mathbf{S}|}\}$ be the volume set containing volumes in $\mathbf{D}^{2(-1)}$ (if $|\mathbf{D}| = N$) or $\mathbf{E}^2 - \mathbf{F}^2 + \mathbf{F}^{2(-1)}$ (if $|\mathbf{D}| \ne N$ and $|\mathbf{E}| = N$), where the volumes in $\mathbf{S}$ are sorted in ascending order. Intuitively, $\mathbf{S}$ is the volume set of elementary range queries or complemented elementary range queries. For $i \in [|\mathbf{S}|]$, do: if $i = 1$, set $s_1 = a_1$; otherwise, set $s_i = a_i - a_{i-1}$. Finally, we have the following two cases:
  - Case 1: $\widehat{r}_1 = 1$ (if $|\mathbf{D}| = N$) or $\widehat{r}_2 = 1$ (if $|\mathbf{D}| \ne N$ and $|\mathbf{E}| = N$). For $i \in [|\mathbf{S}|]$, set $vol([i,i]_q) = s_i$;
  - Case 2: $N+1-\widehat{r}_1 = N$ (if $|\mathbf{D}| = N$) or $N+1-\widehat{r}_2 = N$ (if $|\mathbf{D}| \ne N$ and $|\mathbf{E}| = N$). For $i \in [|\mathbf{S}|]$, set $vol([N+1-i, N+1-i]_q) = s_i$.

  Otherwise (i.e., $|\mathbf{D}| \ne N$ and $|\mathbf{E}| \ne N$), find the smallest volume in $\mathbf{G}^2$, $\mathbf{E}^2$, denoted by x_1, x_2, respectively. Obtain set $\mathbf{S}_1 = \mathbf{G}^2 - \mathbf{F}^{2(-1)}$, sort $\mathbf{S}_1$ in ascending order to obtain a set $\{s_{1,1}, s_{1,2}, ..., s_{1,\widehat{s}_1}\}$, Also, obtain set $\mathbf{S}_2 = \mathbf{E}^2 - \mathbf{F}^2$, sort $\mathbf{S}_2$ in ascending order to obtain a set $\{s_{2,1}, s_{2,2}, ..., s_{2,\widehat{s}_2}\}$. There are four cases:

  Case 1: if $(\widehat{r}_1, \widehat{r}_2)$ is the final result,
  - If $|\widehat{r}_1 - \widehat{r}_2| \ne 1$, obtain $vol([\widehat{r}_1, \widehat{r}_1]_q) = x_1$, $vol([\widehat{r}_2, \widehat{r}_2]_q) = x_2$.
  - If $|\widehat{r}_1 - \widehat{r}_2| = 1$, do as follows:
    * if $\widehat{r}_2 = \widehat{r}_1 + 1$, obtain that $vol([\widehat{r}_1 + 1 - i, \widehat{r}_1]) = s_{1,i}$ for $i \in [\widehat{r}_1]$ and $vol([\widehat{r}_2, i + \widehat{r}_2 - 1]) = s_{2,i}$ for $i \in [N+1-\widehat{r}_2]$. Using the above knowledge, we can obtain other counts of values in $[N]$ as follows: compute $vol([i,i]) = vol([i, \widehat{r}_1]) - vol(i+1, \widehat{r}_1)$ for $i \in [\widehat{r}_1 - 1]$, $vol([\widehat{r}_2 + i, \widehat{r}_2 + i]) = vol([\widehat{r}_2, i + \widehat{r}_2]) - vol(\widehat{r}_2, i + \widehat{r}_2 - 1)$ for $i \in [N - \widehat{r}_2]$.
    * if $\widehat{r}_1 = \widehat{r}_2 + 1$, obtain that $vol([\widehat{r}_1, i + \widehat{r}_1 - 1]) = s_{1,i}$ for $i \in [N+1-\widehat{r}_1]$ and $vol([\widehat{r}_2 + 1 - i, \widehat{r}_2]) = s_{2,i}$ for $i \in [\widehat{r}_2]$. Using the above knowledge, we can obtain other counts of values in $[N]$ as follows: compute $vol([i,i]) = vol([i, \widehat{r}_2]) - vol(i+1, \widehat{r}_2)$ for $i \in [\widehat{r}_2 - 1]$, $vol([\widehat{r}_1 + i, \widehat{r}_1 + i]) = vol([\widehat{r}_1, i + \widehat{r}_1]) - vol(\widehat{r}_1, i + \widehat{r}_1 - 1)$ for $i \in [N - \widehat{r}_1]$.

  Case 2: if $(\widehat{r}_1, N+1-\widehat{r}_2)$ is the final result, obtain $vol([\widehat{r}_1, \widehat{r}_1]_q) = x_1$, $vol([N+1-\widehat{r}_2, N+1-\widehat{r}_2]_q) = x_2$.

  Case 3: if $(N+1-\widehat{r}_1, \widehat{r}_2)$ is the final result, obtain $vol([N+1-\widehat{r}_1, N+1-\widehat{r}_1]_q) = x_1$, $vol([\widehat{r}_2, \widehat{r}_2]_q) = x_2$.

  Case 4: if $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$ is the final result,
  - If $|R+1-\widehat{r}_1 - (N+1-\widehat{r}_2)| \ne 1$, obtain $vol([N+1-\widehat{r}_1, N+1-\widehat{r}_1]_q) = x_1$, $vol([N+1-\widehat{r}_2, N+1-\widehat{r}_2]_q) = x_2$.
  - If $|N+1-\widehat{r}_1 - (N+1-\widehat{r}_2)| = 1$, do as follows:
    * if $N+1-\widehat{r}_2 = (N+1-\widehat{r}_1) + 1$, obtain that $vol([N+2-\widehat{r}_1 - i, N+1-\widehat{r}_1]) = s_{1,i}$ for $i \in [N+1-\widehat{r}_1]$ and $vol([N+1-\widehat{r}_2, i+N-\widehat{r}_2]) = s_{2,i}$ for $i \in [\widehat{r}_2]$. Using the above knowledge, we can obtain other counts of values in $[N]$ as follows: compute $vol([i,i]) = vol([i, N+1-\widehat{r}_1]) - vol(i+1, N+1-\widehat{r}_1)$ for $i \in [N-\widehat{r}_1]$, $vol([N+1-\widehat{r}_2+i, N+1-\widehat{r}_2+i]) = vol([N+1-\widehat{r}_2, i+N+1-\widehat{r}_2]) - vol(N+1-\widehat{r}_2, i+(N+1-\widehat{r}_2)-1)$ for $i \in [\widehat{r}_2 - 1]$.
    * if $N+1-\widehat{r}_1 = (N+1-\widehat{r}_2) + 1$, obtain that $vol([N+1-\widehat{r}_1, i+N-\widehat{r}_1]) = s_{1,i}$ for $i \in [\widehat{r}_1]$ and $vol([N+2-\widehat{r}_2 - i, N+1-\widehat{r}_2]) = s_{2,i}$ for $i \in [N+1-\widehat{r}_2]$. Using the above knowledge, we can obtain other counts of values in $[N]$ as follows: compute $vol([i,i]) = vol([i, N+1-\widehat{r}_2]) - vol(i+1, N+1-\widehat{r}_2)$ for $i \in [N-\widehat{r}_2]$, $vol([N+1-\widehat{r}_1+i, N+1-\widehat{r}_1+i]) = vol([N+1-\widehat{r}_1, i+N+1-\widehat{r}_1]) - vol(N+1-\widehat{r}_1, i+(N+1-\widehat{r}_1)-1)$ for $i \in [\widehat{r}_1 - 1]$.

Example of attack. In the following, we present an example from Fig. 3 to show the process of the attack in Section 4.

- **Step 1.** In this step, we obtain three sets $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, where $\mathbf{A}$ is Knowledge III in Fig. 3, $\mathbf{B}$ is Knowledge VI in Fig. 3 and $\mathbf{C}$ is Knowledge IX in Fig. 3. The size n of $\mathbf{A}$ (or $\mathbf{B}$, or $\mathbf{C}$) is 21. We then obtain the following sets:
  $\mathbf{D} = \mathbf{B} - \mathbf{A} \cap \mathbf{B}$=$\{(b_{1,2}, 5), (b_{1,3}, 20), (b_{1,4}, 25), (b_{1,5}, 28), (b_{1,6}, 31), (b_{2,2}, 3), (b_{2,3}, 18), (b_{2,4}, 23), (b_{2,5}, 26), (b_{2,6}, 29)\}$.
  $\mathbf{E} = \mathbf{B} - \mathbf{B} \cap \mathbf{C} = \{((b_{1,3}, 20), (b_{1,4}, 25), (b_{1,5}, 28), (b_{1,6}, 31), (b_{2,3}, 18), (b_{2,4}, 23), (b_{2,5}, 26), (b_{2,6}, 29), (a_{3,3}, 15), (a_{3,4}, 20), (a_{3,5}, 23), (a_{3,6}, 26)\}$.
  $\mathbf{F} = \mathbf{D} \cap \mathbf{E} = \{(b_{1,3}, 20), (b_{1,4}, 25), (b_{1,5}, 28), (b_{1,6}, 31), (b_{2,3}, 18), (b_{2,4}, 23), (b_{2,5}, 26), (b_{2,6}, 29)\}$.
  $\mathbf{G} = \mathbf{A} - \mathbf{A} \cap \mathbf{B} = \{(a_{1,2}, 4), (a_{1,3}, 19), (a_{1,4}, 24), (a_{1,5}, 27), (a_{1,6}, 30), (a_{2,2}, 2), (a_{2,3}, 17), (a_{2,4}, 22), (a_{2,5}, 25), (a_{2,6}, 28)\}$.
  Hence, we obtain the update-access-volume count for the first update $|\mathbf{D}| = 10$, the update-access-volume count for the second update $|\mathbf{E}| = 12$, the the update-intersection count $|\mathbf{F}| = 8$.
- **Step 2.** Now we know $n = 21$ and $21 = N(N+1)/2$ (from **Observation 4.1**), we can obtain that $N = 6$. Knowing $|\mathbf{D}| = 10$ and $10 = r_1(6+1-r_1)$ (from **Observation 4.3**), we can obtain two values: $\widehat{r}_1 = 2$ or $N+1-\widehat{r}_1 = 5$. Similarly, knowing $|\mathbf{E}| = 12$ and $12 = r_1(6+1-r_1)$ (from **Observation 4.3**), we can obtain two values: $\widehat{r}_2 = 3$ or $N+1-\widehat{r}_2 = 4$. Now, we have four candidate pairs:

5. The case where $r_1 = r_2 = (N+1)/2$ is included.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

$(2,3),(2,4),(5,3),(5,4)$. We further check the four candidate pairs as follows:

- For $(2,3)$, since $2<3$, compute $2(6+1-3)=8$, which equals $|\mathbf{F}|$, so $(2,3)$ is the result.
- For $(2,4)$, compute $2\cdot(6+1-4)=6$, which does not equal $|\mathbf{F}|$, so $(2,4)$ is not the result.
- For $(5,3)$, compute $(6+1-5)\cdot 3=6$, which does not equal $|\mathbf{F}|$, so $(5,3)$ is not the result.
- For $(5,4)$, since $5>4$, compute $4\cdot(6+1-5)=8$, which equals $|\mathbf{F}|$, so $(5,4)$ is the result.

Hence, the final results are $(\widehat{r}_1=2,\widehat{r}_2=3),(N+1-\widehat{r}_1=5,N+1-\widehat{r}_2=4)$.

- **Step 3.** Since $|\mathbf{D}|=10\neq N=6$ and $|\mathbf{E}|=12\neq N=6$, we proceed as follows. Knowing $\mathbf{G}$, $\mathbf{E}$, $\mathbf{F}$, we can obtain $\mathbf{G}^2=\{4,19,24,27,30,2,17,22,25,28\}$, $\mathbf{E}^2=\{20,25,28,31,18,23,26,29,15,20,23,26\}$, $\mathbf{F}^2=\{20,25,28,31,18,23,26,29\}$, $\mathbf{F}^{2(-1)}=\{19,24,27,30,17,22,25,28\}$. We can further obtain $\mathbf{S}_1=\mathbf{G}^2-\mathbf{F}^{2(-1)}=\{4,2\}$, sort $\mathbf{S}_1$ in ascending order to obtain $s_{1,1}=2$, $s_{1,2}=4$. Also, we can obtain $\mathbf{S}_2=\mathbf{E}^2-\mathbf{F}^2=\{15,20,23,26\}$, sort $\mathbf{S}_2$ in ascending order to obtain $s_{2,1}=15$, $s_{2,2}=20$, $s_{2,3}=23$, $s_{2,4}=26$. We have the following two cases:
  - For $(\widehat{r}_1=2,\widehat{r}_2=3)$, since $\widehat{r}_2=\widehat{r}_1+1$, we have that $vol([2,2]_q)=s_{1,1}=2$, $vol([1,2]_q)=s_{1,2}=4$, $vol([3,3]_q)=s_{2,1}=15$, $vol([3,4]_q)=s_{2,2}=20$, $vol([3,5]_q)=s_{2,3}=23$, $vol([3,6]_q)=s_{2,4}=26$. Hence, we have $vol([1,1]_q)=vol([1,2]_q)-vol([2,2]_q)=2$, $vol([4,4]_q)=vol([3,4]_q)-vol([3,3]_q)=5$, $vol([5,5]_q)=vol([3,5]_q)-vol([3,4]_q)=3$, $vol([6,6]_q)=vol([3,6]_q)-vol([3,5]_q)=3$. The counts of values in $[N]$ are $\{2,2,15,5,3,3\}$.
  - For $(N+1-\widehat{r}_1=5,N+1-\widehat{r}_2=4)$, since $N+1-\widehat{r}_1=N+1-\widehat{r}_2+1$, we have that $vol([5,5]_q)=s_{1,1}=2$, $vol([5,6]_q)=s_{1,2}=4$, $vol([4,4]_q)=s_{2,1}=15$, $vol([3,4]_q)=s_{2,2}=20$, $vol([2,4]_q)=s_{2,3}=23$, $vol([1,4]_q)=s_{2,4}=26$. Hence, we have $vol([1,1]_q)=vol([1,4]_q)-vol([2,4]_q)=3$, $vol([2,2]_q)=vol([2,4]_q)-vol([3,4]_q)=3$, $vol([3,3]_q)=vol([3,4]_q)-vol([4,4]_q)=5$, $vol([6,6]_q)=vol([5,6]_q)-vol([5,5]_q)=2$. The counts of values in $[N]$ are $\{3,3,5,15,2,2\}$.

4.6 Explanation and Analysis

Explanation and analysis of Step 1. **Step 1** mainly consists of two procedures, we present the analysis as follows.

- The first procedure is to collect the full-cover access-volume pair set. Different from the attack in Section 3, the attack in this section records the access location for each range query. The analysis regarding the query distribution is the same with that of the attack in Section 3.
- The second procedure is to derive a couple of sets using **A**, **B** and **C**. We explain this procedure as follows. $\mathbf{A}\cap\mathbf{B}$ is the set containing all the access-volume pairs from **A** (or from **B**) that do not change when the first update occurs. Hence, $\mathbf{D}=\mathbf{B}-\mathbf{A}\cap\mathbf{B}$ is the set containing all the access-volume pairs in **B** that are different from any access-volume pair in **A**, $\mathbf{G}=\mathbf{A}-\mathbf{A}\cap\mathbf{B}$ is the set containing all the access-volume pairs in **A** that are different from any access-volume pair in **B**. From **Observation 4.2**, **D** (resp. **G**) is the Type-II (resp. Type-I) update-access-volume set for the first update and $|\mathbf{D}|$ is the (corresponding) update-access-volume count. Similarly, $\mathbf{B}\cap\mathbf{C}$ is the set containing all the access-volume pairs from **B** (or from **C**) that do not change when the second update occurs. Hence, $\mathbf{E}=\mathbf{B}-\mathbf{B}\cap\mathbf{C}$ is the set containing all the access-volume pairs in **B** that are different from any access-volume pair in **C**. From **Observation 4.2**, **E** is the Type-I update-access-volume set for the second update and $|\mathbf{E}|$ is the (corresponding) update-access-volume count. Hence, $\mathbf{F}=\mathbf{D}\cap\mathbf{E}$ is the update-intersection set, $|\mathbf{F}|$ is the update-intersection count. The complexity is $\mathcal{O}(N^2)$.

Explanation and analysis of Step 2. **Step 2** mainly consists of two procedures, we present the analysis as follows.

- The first procedure is to recover N. Since there are $N(N+1)/2$ distinct range queries, from **Observation 4.1**, we can obtain N easily. The complexity is to solve a quadratic equation.
- The second procedure is to compute the values of added records r_1 and r_2. From **Observation 4.3**, we have two possible values for r_1 and two possible values for r_2, up to reflection. Without loss of generality, there are four possible (r_1,r_2) candidate pairs. So the next task is to determine which candidate pair(s) is the result. Based on **Observation 4.5**, with the knowledge of the update-intersection count **F** (obtained from **Step 1**), we can determine which candidate pair(s) is the result.

Explanation and analysis of Step 3. **Step 3** gives the method of recovering the count of value in $[N]$. The first procedure is to check whether $|\mathbf{D}|$ or $|\mathbf{E}|$ equals N, if yes, we can run similar operations as in **Case 2** of **Step 3** in Section 3.4 to reconstruct the database counts. Otherwise, we adopt another approach. Since $\mathbf{G}=\mathbf{A}-\mathbf{A}\cap\mathbf{B}$ and $\mathbf{E}=\mathbf{B}-\mathbf{B}\cap\mathbf{C}$, we have that **G** is the Type-I update-access-volume set for the first update, **E** is the Type-I update-access-volume set for the second update. From **Observation 4.4**, for the case where $|\widehat{r}_1-\widehat{r}_2|\neq 1$, or $(\widehat{r}_1,N+1-\widehat{r}_2)$, or $(N+1-\widehat{r}_1,\widehat{r}_2)$ and $|N+1-\widehat{r}_1-(N+1-\widehat{r}_2)|\neq 1$, we can obtain $vol([r_1,r_1]_q)$ and $vol([r_2,r_2]_q)$ easily; for the case where $|\widehat{r}_1-\widehat{r}_2|=1$ or $|N+1-\widehat{r}_1-(N+1-\widehat{r}_2)|=1$, we can fully reconstruct the counts of the database. The complexity is $\mathcal{O}(N^2)$, which the size of **A** (or **B**, or **C**).

Explanations of observations. Since **Observations 4.1, 4.2, 4.3, 4.6** are easy to understand, we do not provide the analysis here. In the following, we main present the explanation of **Observation 4.4** and **Observation 4.5**, respectively.

- **Explanation of Observation 4.4.** Since $\mathbf{U}^2_{1,1}$ is defined to be the volume set of all range queries that will change when the first update with value r_1 occurs, we have that the range query set corresponding to $\mathbf{U}^2_{1,1}$ is $\{[i,j]_q\}_{i\in[1,r_1],j\in[r_1,N]}$, and $vol([r_1,r_1]_q)$ is the smallest volume. Hence, the smallest volume of $\mathbf{U}^2_{1,1}$ must be the volume of $[r_1,r_1]_q$. Similarly, since $\mathbf{U}^2_{2,1}$ is defined to be the volume set of all range queries that will change when the second update with value r_2 occurs, we have that the range query set corresponding to $\mathbf{U}^2_{2,1}$ is $\{[i,j]_q\}_{i\in[1,r_2],j\in[r_2,N]}$, and $vol([r_2,r_2]_q)$ is the smallest volume. Hence, the smallest volume of $\mathbf{U}^2_{2,1}$ must be the volume of $[r_2,r_2]_q$. $\mathbf{U}_{1,2}\cap\mathbf{U}_{2,1}$ denotes the set containing the access-volume pairs that have been changed after the first update and will change once more when the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

second update occurs. Hence, $\mathbf{S}_1 = \mathbf{U}_{1,2} - \mathbf{U}_{1,2} \cap \mathbf{U}_{2,1}$ is the set containing the access-volume pairs that has been changed after the first update and will not change when the second update occurs, and $\mathbf{S}_2 = \mathbf{U}_{2,1} - \mathbf{U}_{1,2} \cap \mathbf{U}_{2,1}$ is the set containing the access-volume pairs that do not change after the first update and will change when the second update occurs. If $r_2 = r_1 + 1$, the range query set corresponding to $\mathbf{S}_1$ is $\{[i, r_1]_q\}_{i\in[r_1]}$, and the range query set corresponding to $\mathbf{S}_2$ is $\{[r_2, i]_q\}_{i\in[r_2,N]}$. On the other hand, if $r_1 = r_2 + 1$, the range query set corresponding to $\mathbf{S}_1$ is $\{[r_1, i]_q\}_{i\in[r_1,N]}$, and the range query set corresponding to $\mathbf{S}_2$ is $\{[i, r_2]_q\}_{i\in[r_2]}$. Given the volumes of $\{[i, r_1]_q\}_{i\in[r_1]}$ and $\{[r_2, i]_q\}_{i\in[r_2,N]}$, or the volumes of $\{[r_1, i]_q\}_{i\in[r_1,R]}$ and $\{[i, r_2]_q\}_{i\in[r_2]}$, from **Observation 4.6**, it is easy to recover the count of each value in $[N]$.
- **Explanation of Observation 4.5.** Given $\widehat{r}_1$, $N+1-\widehat{r}_1$, $\widehat{r}_2$, $N+1-\widehat{r}_2$ defined in **Observation 4.4**, we analysis the four cases of this observation as follows:
  - If $(\widehat{r}_1, \widehat{r}_2)$ is the result, there are two cases:
    - If $\widehat{r}_1 < \widehat{r}_2$, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,\widehat{r}_1],j\in[\widehat{r}_2,N]}$. Hence, the update-intersection count is $\widehat{r}_1(N+1-\widehat{r}_2)$;
    - If $\widehat{r}_2 < \widehat{r}_1$, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,\widehat{r}_2],j\in[\widehat{r}_1,N]}$. Hence, the update-intersection count is $\widehat{r}_2(N+1-\widehat{r}_1)$.
  - If $(\widehat{r}_1, N+1-\widehat{r}_2)$ is the result, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,\widehat{r}_1],j\in[N+1-\widehat{r}_2,N]}$. Hence, the update-intersection count is $\widehat{r}_1 \cdot \widehat{r}_2$;
  - If $(N+1-\widehat{r}_1, \widehat{r}_2)$ is the result, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,\widehat{r}_2],j\in[N+1-\widehat{r}_1,N]}$. Hence, the update-intersection count is $\widehat{r}_1 \cdot \widehat{r}_2$;
  - If $(N+1-\widehat{r}_1, N+1-\widehat{r}_2)$ is the result, there are two cases:
    - If $N+1-\widehat{r}_1 < N+1-\widehat{r}_2$, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,N+1-\widehat{r}_1],j\in[N+1-\widehat{r}_2,N]}$. Hence, the update-intersection count is $\widehat{r}_2(N+1-\widehat{r}_1)$;
    - If $N+1-\widehat{r}_2 < N+1-\widehat{r}_1$, the range query set corresponding to the update-intersection set is $\{[i, j]_q\}_{i\in[1,N+1-\widehat{r}_2],j\in[N+1-\widehat{r}_1,N]}$. Hence, the update-intersection count is $\widehat{r}_1(N+1-\widehat{r}_2)$.

5 EXTENSIONS TO SUPPORT DELETION

The operation for database update considered in Section 3 and Section 4 is to add new record(s). Deletion is another common form of database update. In this section, we consider the scenario where the update operation is deletion.

5.1 Extension of Attack with One Update

In this section, we present the extension of the attack in Section 3. As in Section 3, here we consider the dense database setting, but after deletion, the database may become non-dense. Let $\mathbf{K}$ be the set containing all the values that *has one count* (i.e., $\mathbf{K} = \{i|vol([i, i]_q) = 1\}_{i\in[N]}$), $\mathbf{A}$ and $\mathbf{B}$ be the full-cover volume set before and after the update of the database, respectively. Let the the value of the deleted record be r, we have the following additional observation.

- **Observation 5.1:** If $|\mathbf{A}| = |\mathbf{B}|$, then $r \notin \mathbf{K}$ and the equation $|\mathbf{B}| = |\mathbf{A}| = N(N+1)/2$ holds; otherwise, if $|\mathbf{A}| = |\mathbf{B}|+N$, then $r \in \mathbf{K}$, and the equation $|\mathbf{A}| = |\mathbf{B}|+N = N(N+1)/2$ holds.

Based on the above additional observation, we consider the attack for two cases as follows.

Case 1: $|\mathbf{A}| = |\mathbf{B}|$. In this case, we extend the attack in Section 3.4 by modifying **Step 1** as follows:

- **Step 1.** This step is almost the same with that of the attack in Section 3.4, except for replacing $b_i - a_i = 1$ with $a_i - b_i = 1$.

Case 2: $|\mathbf{A}| = |\mathbf{B}| + N$. In this case, the attack works as follows:

- **Step 1: Obtain the update-volume count C.** Let $\mathbf{A}$, $\mathbf{B}$ be the full-cover access-volume pair sets before and after the database update, respectively. Obtain $\mathbf{D} = \mathbf{A} - \mathbf{A} \cap \mathbf{B}$ and $C = |\mathbf{D}|$.
- **Step 2: Recover the domain size N and the value of newly added record r.** With the knowledge of $n = |\mathbf{A}|$, obtain N from equation $n = N(N+1)/2$. Then, with the knowledge of C and N, obtain r from equation $C = r(N+1-r)$.
- **Step 3: Recover the count of value in $[N]$.** There are two cases:
  Case 1: $C \neq N$. Find the smallest value in $\mathbf{D}^2$, denoted by a, obtain $vol([r, r]_q) = a$.
  Case 2: $C = N$. Knowing $C = N$, we have that $r = 1$ or $r = N$ from **Step 2**. Let $\mathbf{S} = \{a_1, a_2, ..., a_{|\mathbf{S}|}\}$ be the volume set of all volumes in $\mathbf{D}^2$ in ascending order. Hence, $\mathbf{S}$ is the volume set of elementary range queries or complemented elementary range queries. For $i \in [|\mathbf{S}|]$, do: if $i = 1$, set $s_1 = a_1$; otherwise, set $s_i = a_i - a_{i-1}$. Finally, we have the following two cases:
  - Case 1: $r = 1$. For $i \in [|\mathbf{S}|]$, set $vol([i, i]_q) = s_i$;
  - Case 2: $r = N$. For $i \in [|\mathbf{S}|]$, set $vol([N+1-i, N+1-i]_q) = s_i$.

5.2 Extension of Attack with Two Updates

In this section, we present the extension of the attack in Section 4. As in Section 4, we here consider the dense database setting and the values of the deleted records are different[6], but after deletion, the database may become non-dense. Let $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ be the full-cover access-volume pair sets before $\mathbf{T}_1$, after $\mathbf{T}_1$ but before $\mathbf{T}_2$, and after $\mathbf{T}_2$, respectively. We consider the following two specific cases:

- **Case 1:** The counts of values are all above 2;
- **Case 2:** Only one value has one count (and the counts of other values are all above 1), denoted by k. In addition, the value of the second deleted record is k.

We note that **Case 1** happens with high probability for large-scale database. **Case 2** provides a setting to demonstrate the possibility of reconstruction even when the database becomes non-dense. For the above two cases, we have the following additional observation.

- **Observation 5.2:** If the counts of values are all above 2, the equation $|\mathbf{A}| = |\mathbf{B}| = |\mathbf{C}| = N(N+1)/2$ holds; if

6. For the setting where the values of the two deleted records are identical, it is indeed the same setting as Section 5.1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

$|\mathbf{A}| = |\mathbf{B}| = |\mathbf{C}| + N$, the value of the first deleted record is not k, the value of the second deleted record is k, and the equation $|\mathbf{A}| = |\mathbf{B}| = |\mathbf{C}| + N = N(N+1)/2$ holds.

Based on the above additional observation, for **Case 1**, we can extend the attack in Section 4.5 by modifying **Step 3** as follows:

- **Step 3** is almost the same with that of the attack in Section 4.5, except for replacing (1) $\mathbf{D}^{2(-1)}$ with $\mathbf{D}^{2(+1)}$; (2) $\mathbf{E}^2 - \mathbf{F}^2 + \mathbf{F}^{2(-1)}$ with $\mathbf{E}^2 - \mathbf{F}^2 + \mathbf{F}^{2(+1)}$; (3) $\mathbf{S}_1 = \mathbf{G}^2 - \mathbf{F}^{2(-1)}$ with $\mathbf{S}_1 = \mathbf{G}^2 - \mathbf{F}^{2(+1)}$.

For **Case 2**, note that $\mathbf{E} = \mathbf{B} - \mathbf{B} \cap \mathbf{C}$ will not be influenced by the change of $\mathbf{C}$, even when the database become non-dense after the second deletion. Hence, we can extend the attack in Section 4.5 by modifying **Step 3** as follows:

- **Step 3** is the almost same with that of the attack in Section 4.5, excepting for replacing (1) $\mathbf{D}^{2(-1)}$ with $\mathbf{D}^{2(+1)}$; (2) $\mathbf{E}^2 - \mathbf{F}^2 + \mathbf{F}^{2(-1)}$ with $\mathbf{E}^2 - \mathbf{F}^2 + \mathbf{F}^{2(+1)}$; (3) $\mathbf{S}_1 = \mathbf{G}^2 - \mathbf{F}^{2(-1)}$ with $\mathbf{S}_1 = \mathbf{G}^2 - \mathbf{F}^{2(+1)}$.

6 Experiments

In this section, we present the experimental evaluations of our attacks. Since our attack works for general SE schemes providing range queries, similar to [15], we implemented an algorithm that captures the core function of range queries for encrypted database (as the abstract model of encrypted outsourced database described in Section 2.1). This represents a wide range of SE schemes supporting range queries. We simulate an attacker who has observed the full-cover volume set and full-cover access-volume pair set needed in each attack. The number of issued queries is similar to the theoretical analysis in Section 3.5, which depends on the underlying domain size N. We implemented our attacks in Java to recover the value of added or deleted record and reconstruct the database counts. It takes a few hours to collect the full-cover volume set and full-cover access-volume pair set, and the time of the algorithm in our attack is within one hour (depending on the underlying domain size N).

We use datasets consisting of medical records from the 2012 US government's Health Data from New York State to test our attacks. The attributes we chose include age group, length of stay, severity of illness code, clinical classification software diagnosis code (CCSD Code), drug code and major diagnosis category code (Mdc Code), whose domain sizes range from $N = 5$ to $N = 410$. These are indexed via the medical records that are collected from 145 hospitals. In our experiments, the domain sizes for different attributes are listed as follows: AGE_GROUP: 5, LENGTH_OF_STAY: 139, SEVERITY_ILLNESS: 236, CCSD_CODE: 263, DRUG_CODE: 410 and MDC_CODE: 266. We regard the attack as successful if we can recover the value of added or deleted record or reconstruct the desired database count(s) (up to reflection). We define the success rate as the percentage of updates that could be utilized to correctly recover the value of added record or reconstruct the database counts. The value of each added or deleted record is chosen randomly and independently. We repeat each instance of adding or deleting a record that covers all the values in N and eventually take the average of the results.

TABLE 1 Results of Attack with One Update[1]

	Age Group	Length of Stay	Severity of Illness	CCSD Code	Drug Code	Mdc Code
T1	100%	100%	100%	100%	100%	100%
T2	100%	100%	100%	100%	100%	100%
T3	40%	1.438%	0.847%	0.76%	0.487%	0.75%

[1] T1 denotes the success rate of recovering the value (denoted by r) of newly added record, T2 denotes the success rate of recovering $vol([r,r]_q)$, T3 denotes the success rate of fully recovering the database counts.

TABLE 2 Results of Attack with Two Updates[1]

	Age Group	Length of Stay	Severity of Illness	CCSD Code	Drug Code	Mdc Code
T1	100%	100%	100%	100%	100%	100%
T2	100%	100%	100%	100%	100%	100%
T3	90%	4.285%	2.531%	2.272%	1.46%	2.247%

[1] T1 denotes the success rate of recovering the values (denoted by r_1 and r_2) of two newly added records, T2 denotes the success rate of recovering $vol([r_1,r_1]_q)$ and $vol([r_2,r_2]_q)$, T3 denotes the success rate of fully recovering the database counts.

6.1 Attack with One Update

The results of our attack are summarized in Table 1. The results show that the attack in Section 3 can correctly recover the value of newly added record for all attributes. Let r be the value of newly added record, $vol([r,r]_q)$ can also be correctly recovered for all attributes. The success rate of fully reconstructing the database counts depends on the underlying domain size. When the attribute is age group, the success rate is 40%. This is much higher than other attributes shown in the table. The reason is that smaller domain size has higher probability of the updated value being either the minimum or the maximum value in the range.

6.2 Attack with Two Updates

The evaluation results of our attack in Section 4 are summarized in Table 2. The results show that the values of newly added records can be recovered with 100% for all attributes. Let r_1, r_2 be the values of newly added records ($r_1 \neq r_2$), the success rate of recovering $vol([r_1,r_1]_q)$ and $vol([r_2,r_2]_q)$ is also 100% for all attributes. As with the evaluation in the previous section, the success rate of fully reconstructing the database counts depends on the domain size. Note that the success rate is much higher for the attribute AGE_GROUP than other attributes, which is 90%. This is due to smaller domain size having higher probability of two values that are consecutive numbers or being either the minimum or the maximum value in the range.

6.3 Attack Supporting Deletion

In this section we evaluate the success rate of our attacks in Section 5.1 and Section 5.2. The results are summarized in Table 3 and Table 4, respectively. Table 3 shows that the attack in Section 5.1 can recover the value of newly deleted record with 100% for all attributes. Given r the value of newly deleted record, the success rate of recovering $vol([r,r]_q)$ is also 100% for all attributes. As with the previous case on one update, the success rate of fully reconstructing the database counts depends on the domain size

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

TABLE 3 Results of Attack supporting Deletion with One Update[1]

	Age Group	Length of Stay	Severity of Illness	CCSD Code	Drug Code	Mdc Code
T1	100%	100%	100%	100%	100%	100%
T2	100%	100%	100%	100%	100%	100%
T3	40%	1.438%	0.847%	0.76%	0.487%	0.75%

[1] T1 denotes the success rate of recovering the value (denoted by r) of a deleted record, T2 denotes the success rate of recovering $vol([r, r]_q)$, T3 denotes the success rate of fully recovering the database counts.

TABLE 4 Results of Attack supporting Deletion with Two Update[1]

	Age Group	Length of Stay	Severity of Illness	CCSD Code	Drug Code	Mdc Code
T1	100%	100%	100%	100%	100%	100%
T2	100%	100%	100%	100%	100%	100%
T3	90%	4.285%	2.531%	2.272%	1.46%	2.247%

[1] T1 denotes the success rate of recovering the values (denoted by r_1 and r_2) of two deleted records, T2 denotes the success rate of recovering $vol([r_1, r_1]_q)$ and $vol([r_2, r_2]_q)$, T3 denotes the success rate of fully recovering the database counts.

N, where smaller domain size would have the higher probability that the deleted value is the minimum or maximum value. As can be observed from the table, the success rate for age group is much higher than other attributes. Table 4 further shows that the attack in Section 5.2 can recover the values of deleted records with 100% for all attributes. For our experimental database, the first case of the attack in Section 5.2 is satisfied. As in the attack with two updates, given r_1, r_2 the values of the deleted records, the success rate of recovering $vol([r_1, r_1]_q)$ and $vol([r_2, r_2]_q)$ is 100% for all attributes. The success rate of fully reconstructing the database counts similarly depends on domain size N, as can be seen where the success rate is much higher for the attribute AGE_GROUP than other attributes, which is 90%.

6.4 Comparison

In this section, we present a comparison between our attack in Section 3 and GLMP's attack [15]. We assume that their attack has already obtained the full-cover volume set before the database update, the database counts, as well as the knowledge of the total number of records and the domain size. Also, we assume that in our attack the attacker has obtained the full-cover volume set before and after the database update, respectively. The results are illustrated in Fig. 4. Since the success rate of GLMP's attack heavily relies on the number of records, hence, we compile the number of records at different time slots (i.e., percentage of database) for each attribute in our experiment. This enables us to simulate the success rate between our attack and GLMP's attack for different number of records. We can see that for most of the selected attributes, GLMP's attack cannot recover the value of the added record with 100%[7]. To sum up, in terms of the success rate of recovering the value of the added record, given the full-cover volume set, our attack outperforms GLMP's attack. In addition, GLMP's attack needs to obtain the database counts first, which incurs relatively higher overhead compared to our attack. Nevertheless, GLMP's attack can start with fewer range queries once the database counts are known, but can only approximate the value of the added record with less precision as discussed above.

7. Note that one can modify their attack to recover the value of added record with 100%, but with relatively higher complexity. Specifically, the complex algorithm of their database reconstruction attack must be executed before and after the addition of the new record, respectively. In this case, our attack in Section 3 is much more efficient.

7 Related Work

In addition to the excellent works of Grubbs *et al.* [15] and Kellaris *et al.* [23], there have been many serious attacks proposed on encrypted database utilizing different types of leakage [2], [7], [16], [17], [26], [28], [32], [37]. In the keyword search setting, Cash *et al.* [2] presented leakage-abuse attack for query and plaintext recovery assuming partial or perfect knowledge on distribution of the plaintext. In the setting of active attacker, a related work is the proposal by Zhang *et al.* [39]. They proposed file-injection attacks assuming access pattern leakage. Their attack reveals keyword search queries for both single and conjunctive keyword search. This is related to update recovery attack in that it also operates in the dynamic setting, except that in this case the attacker is allowed to choose and craft the files being injected into the database. Durak *et al.* [8] demonstrated that more information can be extracted from order-revealing encryption ciphertexts than was previously thought. The other recent work on reconstruction attack over range queries is by Lacharité *et al.* [24]. They proposed full reconstruction of database assuming access pattern and ranked leakages without any assumption on query distribution. Assuming only access pattern leakage, they further presented an approximate reconstruction attack. Our work is related but orthogonal to these works in that our goal is on reconstruction based on range queries under the dynamic setting. Recently, Wang *et al.* [36] presented new volume-based attacks on encrypted database. The attacks aim to recover the content of individual user queries by exploiting the behavior of real-world applications, which is different from ours. Akshima *et al.* [1] proposed new attack for multidimensional database from range query access patterns. This is different from ours as we consider one-dimensional database.

Beyond the above attacks on searchable encryption, there have been a series of new constructions that address the problem of information leakage [11]. Faber *et al.* [10] provided the construction supporting rich queries on encrypted data, which includes range, substring, wildcard, and phrase queries. Kamara *et al.* [21] proposed a new structured encryption scheme with only a small amount of information. To enable encrypted rich (range) queries under distributed or outsourced setting, new constructions are proposed by Guo *et al.* [19] and Wu *et al.* [38] respectively. To further reduce information leakage including the knowledge of volume (i.e., the response length) and provide strong security guarantee, new schemes supporting volume-hiding [20], [29] and with rich functionality [31] are recently proposed.

8 Conclusions and Future Work

In this work, we proposed new update recovery attacks that successfully recover the value(s) of the updated record(s)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

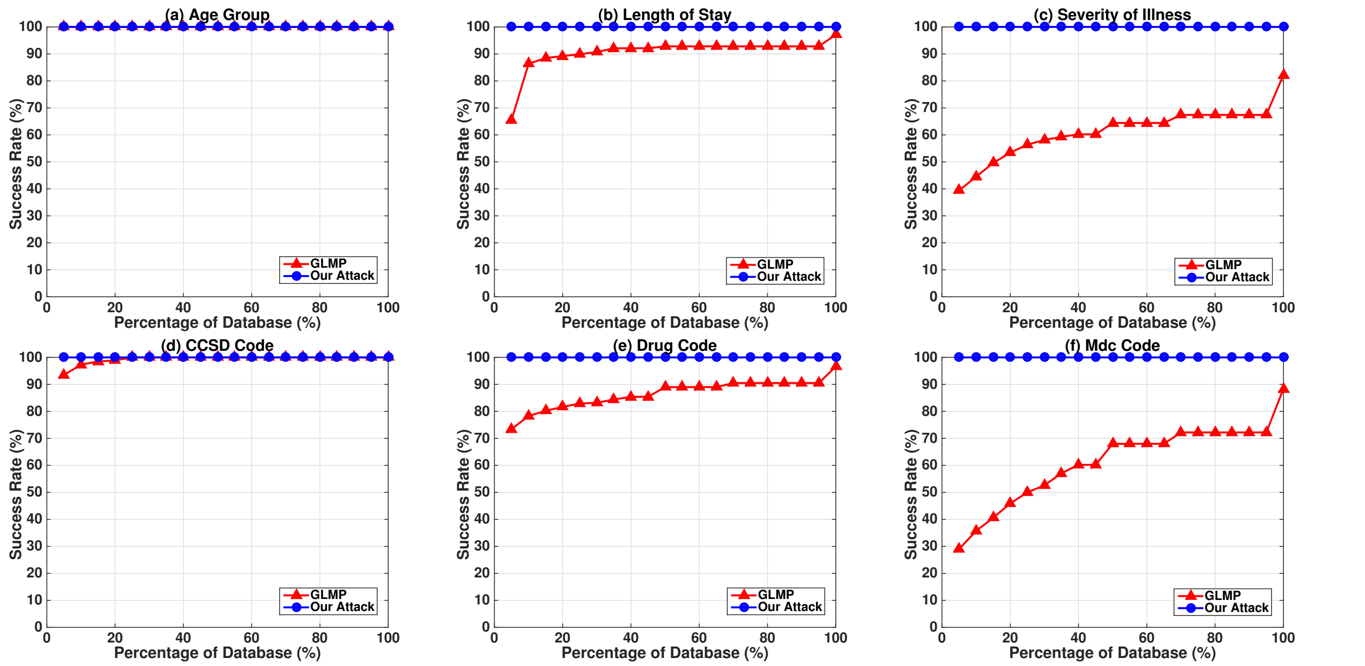


Fig. 4 Comparison between our attack in Section 3 and attack in [15] (denoted by GLMP) in terms of recovering the added record's value.

and enable database reconstruction. We demonstrated that a single database update is sufficient to recover the value of the added record, as well as to fully recover the database counts in two specific cases. In the setting of two updates where two distinct records are added separately, our attack is successful in recovering the values of the added records. The attack can also fully recover the database counts in several cases, which occur with higher probability than the setting of one database update. We further presented update recovery attacks where the update operation is deletion. As far as we know, this has not been considered in attack utilizing volume leakage on encrypted database supporting range queries. Our update recovery attack serves as a reminder that in-depth analysis must be performed on the security impact of leakages when designing dynamic encrypted database schemes. In our future work, we intend to study generalization of our attacks to a more general setting where a series of database updates happen. Similar to GLMP, our attack requires every range query to be issued at least once. In our future work, we will focus on designing new attack that does not suffer from this limitation. Also, our attacks do not work in the scenario where the database is non-dense before any update, whereas an ideal attack should work for the non-dense database setting. We leave the attacks working for non-dense database setting as a future work.

References

[1] Akshima, David Cash, Francesca Falzon, Adam Rivkin, and Jesse Stern. Multidimensional database reconstruction from range query access patterns. *IACR Cryptol. ePrint Arch.*, 2020:296, 2020.

[2] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *ACM CCS 15*, pages 668–679, 2015.

[3] Binyi Chen, Huijia Lin, and Stefano Tessaro. Oblivious parallel ram: improved efficiency and generic constructions. In *TCC (A2) 2016*, pages 205–234. Springer, 2016.

[4] Ronald Cramer, Ivan Damgård, and Yuval Ishai. Share conversion, pseudorandom secret-sharing and applications to secure computation. In *TCC 2005*, pages 342–362, 2005.

[5] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT 2000*, pages 316–334, 2000.

[6] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious ram without random oracles. In *TCC 2011*, pages 144–163. Springer, 2011.

[7] F Betül Durak, Thomas M DuBuisson, and David Cash. What else is revealed by order-revealing encryption? In *ACM CCS 16*, pages 1155–1166. ACM Press, 2016.

[8] F. Betül Durak, Thomas M. DuBuisson, and David Cash. What else is revealed by order-revealing encryption? In *ACM CCS 2016*, pages 1155–1166, 2016.

[9] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *S&P 2012*, pages 332–346. IEEE, 2012.

[10] Sky Faber, Stanislaw Jarecki, Hugo Krawczyk, Quan Nguyen, Marcel-Catalin Rosu, and Michael Steiner. Rich queries on encrypted data: Beyond exact matches. In *ESORICS 2015*, pages 123–145, 2015.

[11] Benjamin Fuller, Mayank Varia, Arkady Yerukhimovich, Emily Shen, Ariel Hamlin, Vijay Gadepally, Richard Shay, John Darby Mitchell, and Robert K. Cunningham. Sok: Cryptographically protected database search. In *S&P 2017*, pages 172–191, 2017.

[12] Craig Gentry. Computing arbitrary functions of encrypted data. *Communications of the ACM*, 53(3):97–105, 2010.

[13] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *ACM STOC*, pages 182–194. ACM Press, 1987.

[14] Paul Grubbs, Anurag Khandelwal, Marie-Sarah Lacharité, Lloyd Brown, Lucy Li, Rachit Agarwal, and Thomas Ristenpart. Pancake: Frequency smoothing for encrypted data stores. Technical report, Technical report, 2020. https://github. com/pancake-security.

[15] Paul Grubbs, Marie-Sarah Lacharite, Brice Minaud, and Kenneth G Paterson. Pump up the volume: Practical database reconstruction from volume leakage on range queries. In *ACM CCS 18*, pages 315–331. ACM Press, 2018.

[16] Paul Grubbs, Richard McPherson, Muhammad Naveed, Thomas Ristenpart, and Vitaly Shmatikov. Breaking web applications built

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

on top of encrypted data. In *ACM CCS 16*, pages 1353–1364. ACM Press, 2016.

[17] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why your encrypted database is not secure. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 162–168. ACM, 2017.

[18] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. Leakage-abuse attacks against order-revealing encryption. In *S&P 2017*, pages 655–672, 2017.

[19] Yu Guo, Xingliang Yuan, Xinyu Wang, Cong Wang, Baochun Li, and Xiaohua Jia. Enabling encrypted rich queries in distributed key-value stores. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1283–1297, 2019.

[20] Seny Kamara and Tarik Moataz. Computationally volume-hiding structured encryption. In *EUROCRYPT 2019*, pages 183–213, 2019.

[21] Seny Kamara, Tarik Moataz, and Olga Ohrimenko. Structured encryption and leakage suppression. In *CRYPTO 2018*, pages 339–370, 2018.

[22] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Accessing data while preserving privacy. In *arXiv 1706.01552*. 2017.

[23] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill. Generic attacks on secure outsourced databases. In *ACM CCS 16*, pages 1329–1340. ACM Press, 2016.

[24] 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, S&P 2018*, pages 297–314, 2018.

[25] Sharad Mehrotra, Shantanu Sharma, Jeffrey D. Ullman, Dhrubajyoti Ghosh, and Peeyush Gupta. Panda: Partitioned data security on outsourced sensitive and non-sensitive data. *CoRR*, abs/2005.06154, 2020.

[26] Muhammad Naveed, Seny Kamara, and Charles V Wright. Inference attacks on property-preserving encrypted databases. In *ACM CCS 15*, pages 644–655. ACM Press, 2015.

[27] Jianting Ning, Jiageng Chen, Kaitai Liang, Joseph K Liu, Chunhua Su, and Qianhong Wu. Efficient encrypted data search with expressive queries and flexible update. *IEEE Transactions on Services Computing*, 2020.

[28] Jianting Ning, Jia Xu, Kaitai Liang, Fan Zhang, and Ee-Chien Chang. Passive attacks against searchable encryption. *IEEE Transactions on Information Forensics and Security*, 14(3):789–802, 2019.

[29] Sarvar Patel, Giuseppe Persiano, Kevin Yeo, and Moti Yung. Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In *ACM CCS 2019*, pages 79–93, 2019.

[30] Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In *Advances in Cryptology - CRYPTO 2010*, pages 502–519, 2010.

[31] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. Arx: an encrypted database using semantically secure encryption. *Proceedings of the VLDB Endowment*, 12(11):1664–1678, 2019.

[32] David Pouliot and Charles V Wright. The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *ACM CCS 16*, pages 1341–1352. ACM Press, 2016.

[33] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *USENIX Security 17*, pages 1357–1374.

[34] Emil Stefanov, Marten Van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: an extremely simple oblivious ram protocol. In *ACM CCS 13*, pages 299–310. ACM Press, 2013.

[35] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *FOCS 2011*, pages 5–16, 2011.

[36] Stephanie Wang, Rishabh Poddar, Jianan Lu, and Raluca Ada Popa. Practical volume-based attacks on encrypted databases. *IACR Cryptol. ePrint Arch.*, 2019:1224, 2019.

[37] Xingchen Wang and Yunlei Zhao. Order-revealing encryption: File-injection attack and forward security. In *ESORICS 2018*, pages 101–121, 2018.

[38] Songrui Wu, Qi Li, Guoliang Li, Dong Yuan, Xingliang Yuan, and Cong Wang. Servedb: Secure, verifiable, and efficient range queries on outsourced database. In *ICDE 2019*, pages 626–637, 2019.

[39] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *USENIX Security 16*, pages 707–720, 2016.

Jianting Ning received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2016. He is currently a research fellow at School of Information Systems, Singapore Management University and a Professor with Fujian Normal University, China. His research interests include applied cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, ESORICS, IEEE TIFS, IEEE TDSC, etc.

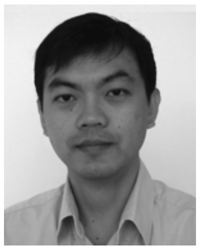
Geong Sen Poh received his PhD degree in Information Security from Royal Holloway, University of London, UK. His main research interests include searchable encryption and cryptographic schemes for computations in the encrypted domain. He was a committee member in the ISO standard for cryptography working group (Malaysia chapter), and committee members for various international conferences. He has published papers in major conferences/journals such as ACM CCS, IEEE TIFS, IEEE TDSC, etc.

Xinyi Huang received his Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published papers in major conferences/journals such as ACM CCS, IEEE TIFS, IEEE TDSC, etc.

Robert H. Deng is AXA Chair Professor of Cybersecurity and Director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests are in the areas of data security and privacy, cloud security and Internet of Things security. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew Fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards and conference committees. These include the editorial boards of IEEE Security & Privacy Magazine, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Information Forensics and Security, Journal of Computer Science and Technology, and Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security. He is an IEEE Fellow.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2020.3015997, IEEE Transactions on Dependable and Secure Computing

Shuwei Cao received the B.Sc and M.Sc. degrees from the School of Computing, National University of Singapore (NUS) in 2016 and 2018, respectively. He is currently a senior research engineer in NUS-Singtel Cybersecurity Lab.



Ee-Chien Chang received the B.Sc and M.Sc. degrees from the National University of Singapore (NUS) and the Ph.D. degree from New York University, New York, in 1998. Currently, he is an associate Professor with the Department of Computer Science, School of Computing, NUS. His research interests include network security and applied cryptography. He has published papers in major conferences/journals such as ACM CCS, USENIX Secuity, ACM SIGMOD, ESORICS, IEEE TIFS, etc.