Strengthening the Security of Encrypted Databases: Non-Transitive JOINs

Ilya Mironov[*] Gil Segev[†*] Ido Shahaf[†]

Abstract

Database management systems operating over encrypted data are gaining significant commercial interest. CryptDB is one such notable system supporting a variety SQL queries over encrypted data (Popa et al., SOSP '11). It is a practical system obtained by utilizing a number of encryption schemes, together with a new cryptographic primitive for supporting SQL's join operator.

This new primitive, an *adjustable join scheme*, is an encoding scheme that enables to generate tokens corresponding to any two database columns for computing their join given only their encodings. Popa et al. presented a framework for modeling the security of adjustable join schemes, but it is not completely clear what types of potential adversarial behavior it captures. Most notably, CryptDB's join operator is *transitive*, and this may reveal a significant amount of sensitive information.

In this work we put forward a strong and intuitive notion of security for adjustable join schemes, and argue that it indeed captures the security of such schemes: We introduce, in addition, natural simulation-based and indistinguishability-based notions (capturing the "minimal leakage" of such schemes), and prove that our notion is positioned between their adaptive and non-adaptive variants.

Then, we construct an adjustable join scheme that satisfies our notion of security based on the linear assumption (or on the seemingly stronger matrix-DDH assumption for improved efficiency) in bilinear groups. Instantiating CryptDB with our scheme strengthens its security by providing a *non-transitive join operator*, while increasing the size of CryptDB's encodings from one group element to four group elements based on the linear assumption (or two group elements based on the matrix-DDH assumption), and increasing the running time of the adjustment operation from that of computing one group exponentiation to that of computing four bilinear maps based on the linear assumption (or two bilinear maps based on the matrix-DDH assumption). Most importantly, however, the most critical and frequent operation underlying our scheme is comparison of single group elements as in CryptDB's join scheme.

[*]Work initiated at Microsoft Research Silicon Valley.

[†]School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: `{segev,ido.shahaf}@cs.huji.ac.il`. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant (Grant No. 618094), by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014632), and by a Google Faculty Research Award.

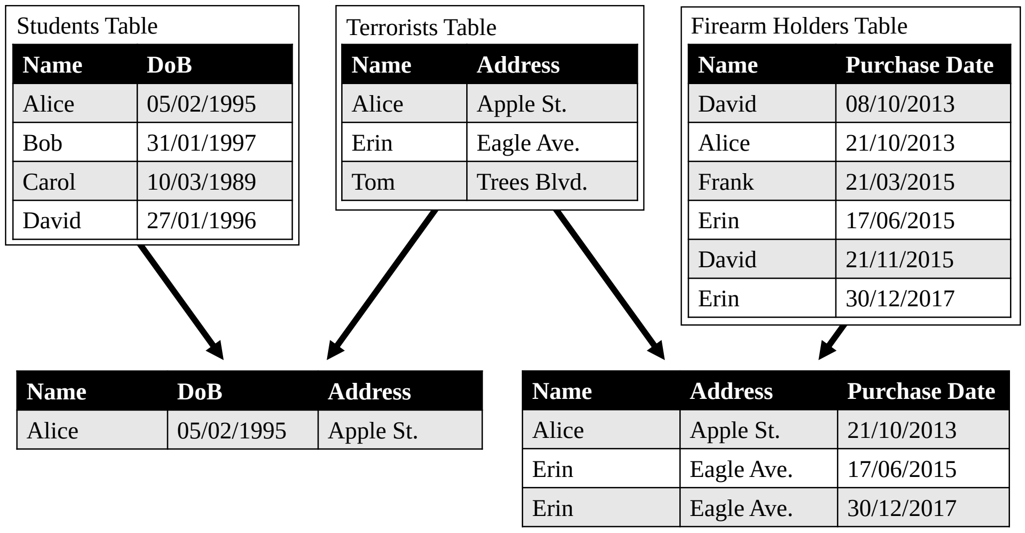
Contents

1 Introduction

Database management systems operating over encrypted data are gaining significant commercial interest. CryptDB, designed by Popa et al. [PRZ$^+$11, PZ12, PRZ$^+$12, Pop14], is one such notable system that supports a variety of SQL queries over encrypted databases. It is a practical system offering a throughput loss of only 26% as compared to MySQL. We refer the reader to CryptDB's project page for the growing list of companies and organizations that have already either adopted CryptDB or designed similar systems directly inspired by CryptDB.[1]

CryptDB operates in a setting that consists of two main parties, a proxy and a server, with the goal of enabling the server to execute SQL queries on encrypted data almost as if it were executing the same queries on the data itself. The only difference is that the operators corresponding to the SQL queries, such as selections, projections, joins, aggregates, and orderings, are performed using possibly modified operators (see, for example, [HIL$^+$02, HIM04, FI12, HKD15, KM16, FVY$^+$17] and the references therein, as well as our discussion in Section 1.3, for additional approaches and systems for executing SQL queries on encrypted data).

Specifically, for our purposes it is sufficient to consider a proxy that holds a secret key sk, and a server that holds a database encrypted using sk. Such a database consists of a number of tables, where each table consists of several data records that are vertically-partitioned into columns. Whenever the proxy would like the server to execute an SQL query, it uses its secret key sk for generating a token allowing the server to execute the given query over the encrypted database. This is realized in CryptDB by utilizing a number of existing encryption schemes, together with a new cryptographic primitive for supporting SQL's join operator (see Figure 1 for a simplified description of SQL's join operator[2]).



Students Table

Name	DoB
Alice	05/02/1995
Bob	31/01/1997
Carol	10/03/1989
David	27/01/1996

Terrorists Table

Name	Address
Alice	Apple St.
Erin	Eagle Ave.
Tom	Trees Blvd.

Firearm Holders Table

Name	Purchase Date
David	08/10/2013
Alice	21/10/2013
Frank	21/03/2015
Erin	17/06/2015
David	21/11/2015
Erin	30/12/2017

Name	DoB	Address
Alice	05/02/1995	Apple St.

Name	Address	Purchase Date
Alice	Apple St.	21/10/2013
Erin	Eagle Ave.	17/06/2015
Erin	Eagle Ave.	30/12/2017

Figure 1: SQL's join operator takes as input two tables, and one or more column labels, and outputs all records that have matching values with respect to the given column labels. There are different types of join operators, depending on the subset of the data records one would like to select from the two given tables. The above example shows the result of joining the "Students" and "Terrorists" tables via their "Name" column, and joining the "Terrorists" and "Firearm Holders" tables via their "Name" column.

[1]CryptDB's project page is available at `css.csail.mit.edu/cryptdb`.

[2]The example described in Figure 1 considers the *inner* join operator, and we note that all of our contributions in this work equally apply to various other join operators, such as right join, left join, full join, and self join.

Adjustable join schemes. Supporting SQL's join operator within CryptDB is essentially equivalent to identifying the matching pairs of values for two encrypted columns, and this has motivated Popa et al. to introduce the notion of an *adjustable join scheme*. This is a symmetric-key encoding scheme supporting the following two operations: (1) Given the secret key sk it is possible to generate an encoding $\mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})$ of any message m relative to any column label col, and (2) given the secret key sk it is possible to generate a token $\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}, \mathsf{col}')$ enabling to compute the join of any two given columns labeled by col and col' (we refer the reader to Section 3 for the formal definition of such schemes). Popa and Zeldovich initiated the study of adjustable join schemes, and presented the first construction of such a scheme, which they have incorporated into the design of CryptDB.

The security of CryptDB's adjustable join. In terms of functionality, a server that is given an encrypted database and a token for computing the join of two columns, should be able to identify all pairs of encodings from these two columns that correspond to identical messages. At the same time, in terms of security, we would like the server not to learn any additional information. Generally speaking, this intuitive requirement can be viewed as a specific instantiation of the security requirement underlying private-key two-input functional encryption (e.g., [GGG+14, BLR+15, BKS16]): Encryption of messages $m_1, \ldots, m_k$ and functional keys corresponding to functions $f_1, \ldots, f_n$ should not reveal any information other than the values $\{f_\ell(m_i, m_j)\}_{i,j \in [k], \ell \in [n]}$.

Popa and Zeldovich [PZ12] formalized a specific notion of security for adjustable join schemes, aiming to capture the above intuitive requirement, and proved that CryptDB's adjustable join scheme indeed satisfies their notion. However, unlike the recently-introduced security notions for private-key functional encryption, it is not completely clear what types of potential adversarial behavior it actually captures.

Most notably, due to efficiency considerations, Popa et al. have chosen to consider a notion of security that does not capture transitivity: For any three columns col_i, col_j and col_k, tokens for computing the joins between col_i and col_k and between col_k and col_j should ideally not allow computing the join between col_i and col_j. Moreover, it is not only that their notion does not capture transitivity, but in fact CryptDB's adjustable join scheme is indeed transitive by design due to efficiency considerations: Given tokens for computing the joins between col_i and col_k and between col_k and col_j, it is easy to compute the join between columns col_i and col_j.

Using our example from Figure 1, this means that given a token for computing the join between the "Students" and "Terrorists" tables (via their "Name" column), and a token for computing the join between the "Terrorists" and "Firearm Holders" tables (again via their "Name" column), the CryptDB server learns that the "Students" and "Firearm Holders" tables have matching records *which were not included in the results of these two join operations* (those matching records belong to David – who is not a terrorist). This may leak significantly more information than one would expect when executing SQL queries over encrypted databases (specifically, in our example, this leaks the fact that among the non-terrorist students there is a student that has two firearms in his or her possession).

In light of the growing commercial interest in CryptDB and in various other similar systems, this state of affairs suggests that a more in-depth security treatment of adjustable join schemes is required, and raises the concrete goal of strengthening the security of CryptDB's adjustable join scheme. Offering a new trade-off between the security of CryptDB and its efficiency is of significant importance especially given the various recent attacks on CryptDB and other similar systems (see, for example, [NKW15, PZB15, GSB+16, KKN+16]).

1.1 Our Contributions

In this work we first put forward a fine-grained definitional framework for adjustable join schemes. Then, we design a new adjustable join scheme for CryptDB that satisfies our strong notions of security, thus offering a new trade-off between the security of CryptDB and its efficiency. In addition, we discusses various extensions of our scheme (e.g., supporting multi-column joins), which can be used for fine-tuning its efficiency, while providing different levels of security, ranging from the security guarantees of CryptDB's join scheme to the stronger security guarantees of our new scheme.

Although our strengthening of CryptDB's security does not directly mitigate the recent attacks on CryptDB (e.g., [NKW15, PZB15]), our new trade-off constitutes a first step towards demonstrating that the security of CryptDB (and, potentially, of other similar systems) can be gradually improved in various aspects. Given the promising applications of such systems and the growing commercial interest in such systems, obtaining a better understanding of such potential trade-offs is an important goal.

We emphasize that an adjustable join scheme is general and system-independent cryptographic primitive. Although our work is motivated by CryptDB, adjustable join schemes can be used by any database system that would like to support join queries over encrypted data, and not only by CryptDB (see, for example, [FI12, HKD15, KM16, FVY$^+$17] and the references therein). Moreover, while our specific construction is designed to be compatible with that of CryptDB, our framework for modeling and defining the security of adjustable join schemes is completely system-independent and is rather likely to find additional applications in various other database systems.

Strengthening the definitional framework. We put forward strong and realistic notions of security for adjustable join schemes, identify the relations among them, and their relations to the notion of security suggested by Popa and Zeldovich [PZ12].

Specifically, we first extend the notion of security considered by Popa and Zeldovich (which we denote by 2Partition) that does not capture transitivity due to efficiency considerations, into a new notion (which we denote by 3Partition) that does capture transitivity. At a first glance, our new notion may still seem rather arbitrary, and it is not immediately clear what types of potential adversarial behaviour it actually captures.

Then, we show that our new notion indeed captures the security of adjustable join schemes: We formalize new simulation-based and indistinguishability-based notions of security, capturing the "minimal leakage" of adjustable join schemes, and prove that 3Partition is positioned between their adaptive variants and non-adaptive variants (i.e., we prove that their adaptive variants imply 3Partition, and that 3Partition implies their non-adaptive variants). We refer the reader to Figure 2 for an illustration of our notions of security and the relations among them, and to Section 1.2 for an overview of our new definitional framework.

Constructing a non-transitive adjustable join scheme. We construct an adjustable join scheme that satisfies our strong notions of security based on the linear assumption [BBS04]. Instantiating CryptDB with our scheme strengthens its security by providing a *non-transitive join operator*, at the expense of increasing the size of CryptDB's encodings from one group element to four group elements, and increasing the running time of the adjustment operation from that of computing one group exponentiation to that of computing four bilinear maps. Most importantly, however, our join operation (which is typically much more frequent than the adjust operation) relies on one comparison of single group elements as in CryptDB.

Moreover, by relying on the seemingly stronger matrix-DDH assumption due to Escala et al. [EHK$^+$17], we obtain a significant improvement to the efficiency of our scheme while still satisfying our strong notion of security. Specifically, basing our scheme on the matrix-DDH assumption results in increasing the size of CryptDB's encodings from one group element to only *two* group elements, and increasing the running time of the adjustment operation from that of computing one group exponentiation to that of computing only *two* bilinear maps (see Section 1.4).

$$\begin{tikzcd}
\text{Adaptive SIM} \arrow[rr, "\text{Claim 4.9}"] \arrow[dr, dashed] & & \text{Adaptive IND} \arrow[dl, "\text{Claim 4.5}"] \\
& \text{3Partition} \arrow[dl, dashed] \arrow[dr, "\text{Claim 4.3}"] & \\
\text{Non-Adaptive SIM} \arrow[rr, leftrightarrow, "\text{Claim 4.7}"] & & \text{Non-Adaptive IND}
\end{tikzcd}$$

Figure 2: An illustration of our notions of security for adjustable join schemes. Solid arrows represent our claims, and dashed arrows follow by transitivity.

1.2 Overview of Our Contributions

In this section we provide a high-level overview of our contributions. First, we briefly describe the notion of an adjustable join scheme. Then, we discuss the notion of security considered by Popa and Zeldovich [PZ12] for such schemes, and CryptDB's transitive join scheme. Finally, we turn to describe our strengthened definitional framework, and the main technical ideas underlying our new scheme.

Adjustable join schemes. As discussed above, an adjustable join scheme [PZ12] is a symmetric-key encoding scheme that enables to generate an encoding $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})$ of any message m relative to any column label col, and to generate a pair of tokens $(\tau, \tau') \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}, \mathsf{col}')$ enabling to compute the join of any two given columns labeled by col and col'. The join is computed publicly via an adjustment algorithm Adj with the following guarantee: For any two column labels col and col' with corresponding tokens $(\tau, \tau') \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}, \mathsf{col}')$, and for any two messages m and m', it holds that

$$m = m' \iff \mathsf{Adj}\left(\tau, \mathsf{Enc}_{\mathsf{sk}}\left(m, \mathsf{col}\right)\right) = \mathsf{Adj}\left(\tau', \mathsf{Enc}_{\mathsf{sk}}\left(m', \mathsf{col}'\right)\right).$$

That is, the scheme adjusts each encoding using its corresponding part of the token, and compares the resulting encodings. It should be noted that we consider schemes that may adjust both columns, whereas CryptDB's scheme adjusts only one of the columns. As far as we can tell, adjusting both columns is fully compatible with the design of CryptDB, and allows for more flexibility when designing adjustable join schemes. We refer the reader to Section 3.1 for a more detailed description of adjustable join schemes.

The security of CryptDB's join scheme. The adjustable join scheme proposed by Popa and Zeldovich [PZ12], as well as the scheme that we proposed in this work, are based on a deterministic encoding algorithm. Clearly, whenever a deterministic encoding algorithm is used, an unavoidable leakage is the equality pattern within each column. When considering, in addition, the functionality of a join scheme, an additional unavoidable leakage is the equality pattern between each pair of columns for which a join token was provided (and this leakage is inherent due to the functionality of the scheme even if the encoding is randomized). However, CryptDB's join scheme leaks significantly more information than the minimal leakage, and our goal is to avoid any unnecessary leakage (as will be formally captured by our notions of security).

Specifically, the notion of security introduced by Popa and Zeldovich, that we denote by $\mathsf{2Partition}$, considers an experiment in which an adversary may adaptively define two disjoint sets of columns, which we refer to as a "left" set L and a "right" set R. The adversary is given the ability to compute joins inside L and joins inside R, but it should not be able to compute the join between any column in L and any column in R.

However, this intuitive requirement does not capture transitivity: Assume that there is a certain column col^* that does not belong to either L or R, then the ability to compute the join between col^* and columns in L, and to compute the join between col^* and columns in R, may imply the ability to compute the join between columns in L and columns in R. Moreover, it is not only that $\mathsf{2Partition}$ does not capture transitivity, but in fact the adjustable join scheme of Popa and Zeldovich (that satisfies $\mathsf{2Partition}$) is indeed transitive due to efficiency considerations (recall our example based on Figure 1). We refer the reader to Section 3.2 for a more detailed discussion of the $\mathsf{2Partition}$ notion and of the adjustable join scheme of Popa and Zeldovich.

Our definitional framework. As our first step, we introduce a new notion of security, denoted $\mathsf{3Partition}$, which strictly extends $\mathsf{2Partition}$. Our notion considers a partitioning of the columns into *three* disjoint sets in a manner that enables it to properly model non-transitive joins. Specifically, we consider adversaries that may adaptively define three disjoint sets of columns, which we refer to as a "left" set L, a "right" set R, and a "middle" set M. The adversary is given the ability to compute joins inside L, inside M, and inside R, as well as joins between L and M and between R and M, but it should not be able to compute the join between any column in L and any column in R.

Intuitively, partitioning the columns into three disjoint sets is inherent when attacking an adjustable join scheme. Consider, for example, a natural security notion asking that an adversary should not be able to distinguish encodings of two databases even when given tokens for computing joins (clearly, this only makes sense as long as the actual results of the join operations do not trivially distinguish the two databases). Then, we claim that the difference between the two databases can be gradually divided into small "changes", each of them implicitly defines a partition into three disjoint sets: There is the set of columns that contain this change, the set of columns that are joined with those columns (and are thus limited to not reveal the difference), and the set of all other columns (which are not subject to any restrictions).

At this point one may ask whether partitioning the columns into three disjoint sets is sufficient for capturing the security of adjustable join schemes, or whether we should also consider partitioning the columns into more than three sets. We show that partitioning the columns into three disjoint sets is sufficient, and that $\mathsf{3Partition}$ indeed captures the security of adjustable join schemes: We formalize new simulation-based and indistinguishability-based notions of security, capturing the "minimal leakage" of adjustable join schemes, and prove that $\mathsf{3Partition}$ in positioned between their adaptive variants and non-adaptive variants (recall Figure 2 for an illustration of our notions of security and the relations among them). We refer the reader to Section 4 for a detailed description and analysis of our definitional framework.

Our adjustable join scheme. Our scheme is inspired by that of Popa et al. [PRZ+11, PZ12, PRZ+12, Pop14]. Their scheme uses a group $\mathbb{G}$ of prime order p that is generated by an element $g \in \mathbb{G}$, and a pseudorandom function for identifying messages and column labels as pseudorandom $\mathbb{Z}_p$ elements. The encoding of a message m for a column col is the group element $g^{a_{\mathsf{col}} \cdot x_m} \in \mathbb{G}$, where

a_{col} and x_m are the pseudorandom $\mathbb{Z}_p$ elements corresponding to col and m, respectively. A token for computing the join between columns col_i and col_j is the element $\tau_{i,j} = a_{\mathsf{col}_i} \cdot a_{\mathsf{col}_j}^{-1} \in \mathbb{Z}_p$, and thus it is clear that such tokens enable transitive joins: Given the tokens $\tau_{i,k} = a_{\mathsf{col}_i} \cdot a_{\mathsf{col}_k}^{-1} \in \mathbb{Z}_p$ and $\tau_{k,j} = a_{\mathsf{col}_k} \cdot a_{\mathsf{col}_j}^{-1} \in \mathbb{Z}_p$, one can efficiently compute the token $\tau_{i,j} = \tau_{i,k} \cdot \tau_{k,j}^{-1}$.

The main idea underlying our scheme is to introduce additional structure into both the encodings and the tokens, and to rely on a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ for computing the adjusted encodings. First, instead of applying a pseudorandom function for identifying messages and column labels as pseudorandom $\mathbb{Z}_p$ elements, we apply a pseudorandom function for identifying messages as pseudorandom $\mathbb{Z}_p^4$ vectors, and column labels as pseudorandom invertible $\mathbb{Z}_p^{4\times 4}$ matrices. In what follows, for a matrix $A = [a_{ij}] \in \mathbb{Z}_p^{a\times b}$ we define $g^A = [g^{a_{ij}}]_{ij} \in \mathbb{G}^{a\times b}$, and for matrices $H = [h_{ij}] \in \mathbb{G}^{a\times b}$ and $H' = [h'_{ij}] \in \mathbb{G}^{b\times c}$ we define $\hat{e}(H, H') = [\prod_{k=1}^{b} \hat{e}(h_{ik}, h'_{kj})]_{ij} \in \mathbb{G}_T^{a\times c}$ (thus, for matrices A and B of appropriate dimensions it holds that $\hat{e}(g^A, g^B) = \hat{e}(g, g)^{AB}$).

Equipped with this notation, the encoding of a message m for a column label col is defined as $c = g^{A_{\mathsf{col}} x_m} \in \mathbb{G}^4$, where $x_m \in \mathbb{Z}_p^4$ and $A_{\mathsf{col}} \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ are the pseudorandom vector and matrix associated with m and col, respectively. Our token-generation algorithm takes as input two column labels, col_i and col_j, uniformly samples a vector $v \leftarrow \mathbb{Z}_p^4$, and outputs the adjustment tokens $g^{v^\mathsf{T} A_{\mathsf{col}_i}^{-1}} \in \mathbb{G}^4$ and $g^{v^\mathsf{T} A_{\mathsf{col}_j}^{-1}} \in \mathbb{G}^4$. Adjusting an encoding $c \in \mathbb{G}^4$ using a token $\tau \in \mathbb{G}^4$ is computed as $\hat{e}(\tau^\mathsf{T}, c) \in \mathbb{G}_T$, and we prove that correctness holds with an overwhelming probability: For any two column labels col_i and col_j, and for any two messages m_i and m_j, it holds that $m_i = m_j$ if and only if $\hat{e}(g^{v^\mathsf{T} A_{\mathsf{col}_i}^{-1}}, g^{A_{\mathsf{col}_i} x_{m_i}}) = \hat{e}(g^{v^\mathsf{T} A_{\mathsf{col}_j}^{-1}}, g^{A_{\mathsf{col}_j} x_{m_j}})$ with all but a negligible probability. We refer the reader to Section 5.1 for the formal description of our scheme.

One may wonder why we use matrices and vectors instead of scalars (as in [PZ12], as well as in [FI12, HKD15]). Otherwise, the scheme is trivially broken, because of the presence of a bilinear map unless rather non-standard assumptions are made (such as the new assumption introduced by Furukawa and Isshiki [FI12] for the purpose of their analysis). In particular, for distinct messages m, m', m'' and columns $\mathsf{col}_i, \mathsf{col}_j$, an adversary can distinguish between $(g^{a_{\mathsf{col}_i} x_m}, g^{a_{\mathsf{col}_i} x_{m'}}, g^{a_{\mathsf{col}_j} x_m}, g^{a_{\mathsf{col}_j} x_{m'}})$ and $(g^{a_{\mathsf{col}_i} x_m}, g^{a_{\mathsf{col}_i} x_{m'}}, g^{a_{\mathsf{col}_j} x_m}, g^{a_{\mathsf{col}_j} x_{m''}})$ by comparing the bilinear image of the first and fourth elements to the bilinear image of the second and third elements.

Our proof of security. For proving the security of our scheme, we first observe that the linear assumption [BBS04] implies that the two distributions $(g^A, g^{Ax}, g^B, g^{By})$ and $(g^A, g^{Ax}, g^B, g^{Bx})$ are computationally indistinguishable, where $A, B \leftarrow \mathbb{Z}_p^{4\times 4}$ and $x, y \leftarrow \mathbb{Z}_p^4$. Intuitively, this enables us to view A and B as $\mathbb{Z}_p^{4\times 4}$ matrices corresponding to two different column labels, and x and y as $\mathbb{Z}_p^4$ vectors corresponding to two different messages. Without being explicitly given a token for computing the join between the columns A and B, an adversary should not be able to distinguish between an encoding g^{Bx} of x to the column B and an encoding g^{By} of y to the column B, even when given an encoding g^{Ax} of x to the column A in both cases.

Our proof of security realizes this intuition, showing that given $(g^A, g^{Ax}, g^B, g^{By})$ or $(g^A, g^{Ax}, g^B, g^{Bx})$ as input we can essentially generate an entire encoding of an adversarially-chosen database, as well as generate all join tokens of the adversary's choice, as long as no token is requested for the join of A and B. Most importantly, although we do not explicitly know either A or B, for any column C we can generate tokens for computing the join between A and C, and the join between C and B. The main challenge, however, is that in our $\mathsf{3Partition}$ notion, the adversary is not limited to only one such pair A and B, and more generally, we do not know in advance the entire structure of the database or the pairs of columns for which the adversary will request join tokens.

Recall that our $\mathsf{3Partition}$ notion considers adversaries that may adaptively define three disjoint sets of columns, which we refer to as a “left” set L, a “right” set R, and a “middle” set M. The adversary is given the ability to compute joins inside L, inside M, and inside R, as well as joins between L and M and between R and M, but it is not given the ability to compute the join between any column in L and any column in R. We rely on this structure for reusing the matrix A for all columns in L and for reusing the matrix B for all columns in R, where in both cases this is done via an appropriate re-randomization. The fact that the adversary is not allowed to request join tokens between L and R guarantees that we are able to generate all required join tokens. We refer the reader to Section 5.2 for our proof of security.

1.3 Additional Related Work

Supporting join queries over encrypted data. Additional approaches for supporting join queries over encrypted data include those of Furukawa and Isshiki [FI12], Hang, Kerschbaum and Damiani [HKD15], and Kamara and Moataz [KM16] which we now discuss.

Furukawa and Isshiki [FI12] consider a notion of security for join schemes which is seemingly weaker compared to our $\mathsf{3Partition}$ notion, and captures non-transitivity to a certain extent (it is essentially equivalent to a non-adaptive variant of our $\mathsf{3Partition}$ notion). However, as we pointed out in Section 1.1, without also including more standard indistinguishability-based and simulation-based notions (as we do in our work), it is far from being clear that such a notion indeed captures the security of join schemes. Furukawa and Isshiki also propose a specific scheme that can be viewed as based on a simplified variant of our scheme where scalars are used instead of matrices and vectors. As discussed in Section 1.2, such a scheme is trivially insecure with respect to our notions of security unless rather non-standard assumptions are made (specifically, Furukawa and Isshiki introduced a new and non-standard assumption for the purpose of their analysis).

Hang, Kerschbaum and Damiani [HKD15] frame their approach in terms of deterministic proxy re-encryption. However, they propose a weak notion of security which does not seem to capture non-transitivity, and their proposed scheme does not satisfy any of our notions of security (or even the notion of security considered by Popa and Zeldovich [PZ12]) under any assumption. As far as we can tell, our scheme is fully compatible with their approach and design goals.

Kamara and Moataz [KM16] recently proposed the first solution for supporting SQL queries on encrypted databases that does not make use of deterministic encodings of the data. Their approach avoids the usage of property-preserving encryption techniques (that are known to be susceptible to various attacks [NKW15, GSB$^+$16, KKN$^+$16]) and of general-purpose primitives such as fully-homomorphic encryption or oblivious RAM (that are currently somewhat unlikely to lead to practical schemes). Their scheme provides strong security guarantees, and in particular a non-transitive join operator. However, their scheme is based on essentially computing all possible joins in advance, and then the problem can be solved via symmetric searchable encryption techniques. Thus, their approach both requires a significant amount of storage (may be quadratic in the size of the database – and thus potentially impractical), and does not seem to support dynamic updates to either the structure or the content of the database.

Proxy re-encryption schemes. Proxy re-encryption schemes (e.g., [BBS98, ID03, AFG$^+$06]) have various applications to distributed storage systems. However, the known constructions and notions of security for proxy re-encryption typically focus on randomized schemes, and therefore (in general) even after invoking the re-encryption algorithm it is not directly clear how to compare two encrypted messages without providing a decryption key – which results in a transitive scheme. Deterministic variants of proxy re-encryption may support such comparisons, as suggested by Hang, Kerschbaum and Damiani [HKD15] and discussed above.

Private set intersection. Adjustable join schemes are somewhat related to the classic problem of designing private set-intersection protocols both in terms of techniques and in terms of security notions. However, in the setting of adjustable join schemes all elements are encoded using a shared secret key sk, whereas in the setting of private set-intersection protocols the parties are not assumed to share any secrets. Moreover, the approach underlying the existing practical protocols does not seem to rely on establishing shared secrets as part of the protocol (see, for example, [HEK12, PSS+15, FHN+16] and the references therein).

Searchable encryption. Adjustable join schemes may also seem somewhat related to symmetric searchable encryption [SWP00, Goh03, CM05, CGK+06, CK10, vLSD+10, CGK+11, KO12, KPR12, CJJ+13, KO13, KP13, BHJ+14, CJJ+14, CT14, CGP+15, ANS+16]. However, in the setting of symmetric searchable encryption a search token is associated with a specific message and enables to identify encryptions of that message, whereas in an adjustable join schemes a join token enables to reveal the equality pattern between two sets of encryptions. As a result, both our notions of security and our techniques are significantly different from those of symmetric searchable encryption. Nevertheless, it would be intriguing to explore any potential applications of our techniques to symmetric searchable encryption.

1.4 Extensions and Open Problems

Multi-column joins. Following the work of Popa et al. [PRZ+11, PRZ+12, Pop14] we have considered joins according to two columns (and thus two tables). Our adjustable join scheme can in fact be extended to support multi-column joins by modifying its token-generation algorithm (and without modifying its encoding or adjustment algorithms). This enables to join multiple tables more efficiently (compared to successively applying two-column joins), and leads to reducing the space overhead by using a smaller number of adjusted encodings.

Specifically, our token-generation algorithm can be modified as follows. On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, $\mathsf{sk} = (K_1, K_2)$ and an arbitrary number $k = k(\lambda)$ column labels $\mathsf{col}_1, \ldots, \mathsf{col}_k \in \mathcal{L}_\lambda$, the modified token-generation algorithm uniformly samples $v \leftarrow \mathbb{Z}_p^4 \setminus \{(0,0,0,0)\}$, computes $A_{\mathsf{col}_i} = \mathsf{PRF}_{K_2}(\mathsf{col}_i) \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ for every $i \in [k]$, and outputs the tuple $\left(g^{v^\mathsf{T} A_{\mathsf{col}_1}^{-1}}, \ldots, g^{v^\mathsf{T} A_{\mathsf{col}_k}^{-1}}\right) \in \left(\mathbb{G}^4\right)^k$ of adjustment tokens. Moreover, we can generate such a multi-join adjustment tokens even when not all k columns are known in advance, as long as we securely store the value v (or, possibly, regenerate it using a pseudorandom function), then compute the value $g^{v^\mathsf{T} A_{\mathsf{col}_i}^{-1}}$ only when col_i is determined.

This allows to reduce space usage (and time as well), by storing adjusted encodings for cliques of joined columns, instead of storing adjusted encodings for each pair of them. Moreover, this allows tuning a trade-off between privacy and efficiency in space and time, by using multiple-column join encodings for the less sensitive data. At the extreme end, one can store only one column of adjusted encodings for each column, by using multiple-column encodings for disjoint sets of columns, and obtain security guarantees that are similar to the $\mathsf{2Partition}$-security of [PZ12]. Overall, our support for multi-column joins enables to fine-tune the efficiency of our scheme, while providing different levels of security, ranging from the security guarantees of CryptDB's join scheme to the stronger security guarantees of our new scheme.

Improved efficiency via the matrix-DDH assumption. The security of our adjustable join scheme is based on the assumption the two distributions $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{By})$ and $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{Bx})$ are computationally indistinguishable, where $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, $A, B \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and $x, y \leftarrow \mathbb{Z}_p^4$. This assumption is the reason that we increase the size of

CryptDB's encodings from one group element to four group elements, and increase the running time of the adjustment operation from that of computing one group exponentiation to that of computing four bilinear maps.

Claim 2.5 (which we prove in Appendix A.1) states that this assumption is implied by the linear assumption [BBS04]. The seemingly stronger $\mathcal{U}_{4,2}$-MDDH assumption due to Escala et al. [EHK+17], states that our underlying assumption holds already for 2×2 matrices instead of 4×4 matrices. In turn, based on the $\mathcal{U}_{4,2}$-MDDH assumption we obtain a more efficient scheme, increasing the size of CryptDB's encodings from one group element to two group elements, and increasing the running time of the adjustment operation from that of computing one group exponentiation to that of computing two bilinear maps. This more efficient scheme is directly obtained from our scheme by simply modifying the dimensions of all 4×4 matrices and 4×1 vectors to dimensions 2×2 and 2×1, respectively (and without any additional modification to either the construction or the proof of security).

Adaptive security. Our adjustable join scheme satisfies our strong $\mathsf{3Partition}$ notion, which considers adversaries that may determine databases of any polynomial size (i.e., databases containing any polynomial number columns and records). When considering databases with a logarithmic number of columns (but still allowing any polynomial number of records!), it is possible to prove that our scheme satisfies our even stronger, adaptive, indistinguishability-based notion.

This is done by "guessing" the partitioning of the columns into three disjoint sets, as implicitly defined by the adversary's token-generation queries within our adaptive indistinguishability experiment – thus leading to only a polynomial security loss as the number of such partitions is polynomial assuming that the number of columns is logarithmic (a rather standard argument shows that the notion of security obtained from $\mathsf{3Partition}$ by not asking the adversary to explicitly partition the columns into three sets, is equivalent to our adaptive indistinguishability-based notion). Similarly, by relying on the standard sub-exponential variant of the linear assumption, we can prove adaptive indistinguishability-based security for databases with any a-priori bounded polynomial number of columns (without requiring any a-priori bound on the polynomial number of records). We leave the task of formalizing these intuitions to future work. An intriguing open problem is to achieve such a level of security without relying on sub-exponential assumptions or without an a-priori bound on the number of columns.

Deterministic vs. randomized encodings. Our encoding algorithm is deterministic similarly and in compatibility with that of CryptDB. An intriguing open problem is to explore the possibility and the potential advantages of join schemes that are based on a randomized encoding algorithm. Given the inherent leakage of the join operation itself, it is not immediately clear that using a randomized encoding algorithm may offer any clear advantage except for avoiding the inherent leakage of deterministic encoding (i.e., the equality pattern within each column).

1.5 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce the basic tools and computational assumptions underlying our contributions. In Section 3 we present the notion of an adjustable join scheme, and discuss the weakness of the notion of security for such schemes that was put forward by Popa and Zeldovich [PZ12]. In Section 4 we introduce our new and refined framework for capturing the security of adjustable join schemes. In Section 5 we present our new adjustable join scheme and prove its security.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$, and for two random variables X and Y we denote by $\Delta(X, Y)$ their statistical distance. The following two facts follow directly from notion of statistical distance:

Fact 2.1. *Let X and Y be two random variables over Ω. Then, for any (possibly randomized) function $f : \Omega \to \Omega'$ it holds that $\Delta(f(X), f(Y)) \leq \Delta(X, Y)$.*

Fact 2.2. *Let X and Y be a random variables over Ω such that $\Pr[Y = \omega] = \Pr[X = \omega \mid A^c]$ for some event A and for all $\omega \in \Omega$. Then, it holds that $\Delta(X, Y) \leq \Pr[A]$.*

Throughout the paper, we denote by $\lambda \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\nu(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically indistinguishable* (denoted $X \approx_s Y$) if $\Delta(X_\lambda, Y_\lambda)$ is negligible in λ. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *computationally indistinguishable* (denoted $X \approx_c Y$) if for any probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that $\left|\Pr_{x \leftarrow X_\lambda}[\mathcal{A}(1^\lambda, x) = 1] - \Pr_{y \leftarrow Y_\lambda}[\mathcal{A}(1^\lambda, y) = 1]\right|$ is negligible in λ. The following fact follows directly from notion of computational indistinguishability:

Fact 2.3. *Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ be computationally indistinguishable. Then, for any probabilistic polynomial-time algorithm $\mathcal{A}$ it holds that $\mathcal{A}(X)$ and $\mathcal{A}(Y)$ are computationally indistinguishable.*

2.1 Pseudorandom Functions

Let $\{\mathcal{K}_\lambda, \mathcal{X}_\lambda, \mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of sets and let $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ be a function family with the following syntax:

- $\mathsf{PRF.Gen}$ is a probabilistic polynomial-time algorithm that takes as input the unary representation of the security parameter λ, and outputs a key $K \in \mathcal{K}_\lambda$.
- $\mathsf{PRF.Eval}$ is a deterministic polynomial-time algorithm that takes as input a key $K \in \mathcal{K}_\lambda$ and a value $x \in \mathcal{X}_\lambda$, and outputs a value $y \in \mathcal{Y}_\lambda$.

The sets $\mathcal{K}_\lambda$, $\mathcal{X}_\lambda$, and $\mathcal{Y}_\lambda$ are referred to as the *key space*, *domain*, and *range* of the function family, respectively. For ease of notation we may denote by $\mathsf{PRF.Eval}_K(\cdot)$ or $\mathsf{PRF}_K(\cdot)$ the function $\mathsf{PRF.Eval}(K, \cdot)$ for $K \in \mathcal{K}_\lambda$. The following is the standard definition of a pseudorandom function family.

Definition 2.4. A function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ is *pseudorandom* if for every probabilistic algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\begin{aligned}\mathsf{Adv}_{\mathsf{PRF},\mathcal{A}}(\lambda) &\stackrel{\mathsf{def}}{=} \left| \Pr_{K \leftarrow \mathsf{PRF.Gen}(1^\lambda)}\left[\mathcal{A}^{\mathsf{PRF.Eval}_K(\cdot)}(1^\lambda) = 1\right] - \Pr_{f \leftarrow F_\lambda}\left[\mathcal{A}^{f(\cdot)}(1^\lambda) = 1\right]\right| \\ &\leq \nu(\lambda),\end{aligned}$$

for all sufficiently large $\lambda \in \mathbb{N}$, where F_λ is the set of all functions that map $\mathcal{X}_\lambda$ into $\mathcal{Y}_\lambda$.

2.2 Computational Assumptions

Let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that takes as input the security parameter 1^λ, and outputs a tuple $(\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, where p is a λ-bit prime number, $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order p, g is a generator of $\mathbb{G}$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate efficiently-computable bilinear map.

For a matrix $A = [a_{ij}] \in \mathbb{Z}_p^{a\times b}$ we define $g^A = [g^{a_{ij}}]_{ij} \in \mathbb{G}^{a\times b}$, and for matrices $H = [h_{ij}] \in \mathbb{G}^{a\times b}$ and $H' = [h'_{ij}] \in \mathbb{G}^{b\times c}$ we define $\hat{e}(H, H') = [\prod_{k=1}^{b} \hat{e}(h_{ik}, h'_{kj})]_{ij} \in \mathbb{G}_T^{a\times c}$ (thus, for matrices A and B of appropriate dimensions it holds that $\hat{e}(g^A, g^B) = \hat{e}(g, g)^{AB}$). We denote by $\mathsf{Rk}_r(\mathbb{Z}_p^{a\times b})$ the set of all $a \times b$ matrices over $\mathbb{Z}_p$ of rank r.

The linear assumption [BBS04] states that for $\mathsf{params} \leftarrow \mathcal{G}(1^\lambda)$, $g_1, g_2, g_3 \leftarrow \mathbb{G}$ and $r_1, r_2, r_3 \leftarrow \mathbb{Z}_p$, the two distributions $(\mathsf{params}, g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_3})$ and $(\mathsf{params}, g_1, g_2, g_3, g_1^{r_1}, g_2^{r_2}, g_3^{r_1+r_2})$ are computationally indistinguishable. The security of our scheme relies on the following assumption – which we prove to follow from the linear assumption (see Appendix A.1):

Claim 2.5. *The linear assumption implies that the two distributions* $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{By})$ *and* $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{Bx})$ *are computationally indistinguishable, where* $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$*,* $A, B \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ *and* $x, y \leftarrow \mathbb{Z}_p^4$*.*

As discussed in Section 1.4, a variant of the above claim for 2×2 matrices is implied by the matrix-DDH assumption due to Escala et al. [EHK+17]. Specifically, their $\mathcal{U}_{4,2}$-MDDH assumption states that the two distributions $(\mathsf{params}, g^C, g^{Cv})$ and $(\mathsf{params}, g^C, g^u)$ are computationally indistinguishable, where $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, $C \leftarrow \mathbb{Z}_p^{4\times 2}$, $v \leftarrow \mathbb{Z}_p^2$ and $u \leftarrow \mathbb{Z}_p^4$.

Claim 2.6. *The* $\mathcal{U}_{4,2}$*-MDDH assumption implies that the two distributions* $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{By})$ *and* $(\mathsf{params}, g^A, g^{Ax}, g^B, g^{Bx})$ *are computationally indistinguishable, where* $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$*,* $A, B \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{2\times 2})$ *and* $x, y \leftarrow \mathbb{Z}_p^2$*.*

The proof of Claim 2.6 is rather straightforward. Given $(\mathsf{params}, g^C, g^u)$, where either $u \leftarrow \mathbb{Z}_p^4$ or $u = Cv$ where $v \leftarrow \mathbb{Z}_p^2$, we view C and u as consisting of two equal-sized matrices and vectors, respectively,

$$C = \begin{bmatrix} A \\ B \end{bmatrix}, \quad u = \begin{bmatrix} w \\ z \end{bmatrix},$$

and rearrange the tuple as $(\mathsf{params}, g^A, g^w, g^B, g^z)$. Since the probability that A or B are not invertible is negligible, by Fact 2.2 we may assume that they are invertible. Now, for $x, y \leftarrow \mathbb{Z}_p^2$, it holds that Ax and By are independent, uniformly distributed, and independent of A and B. So, if $u \leftarrow \mathbb{Z}_p^4$, then w and z are distributed as Ax and By. On the other hand, if $u = Cv$ where $v \leftarrow \mathbb{Z}_p^2$, then $w = Av$ and $z = Bv$. So, distinguishing between the two ensembles in Claim 2.6 would result in contradicting the $\mathcal{U}_{4,2}$-MDDH assumption.

3 Adjustable Join Schemes and Their Security

In this section we first present the notion of an adjustable join scheme [PRZ+11, PZ12, PRZ+12, Pop14]. Then, we present the notion of security introduced by Popa and Zeldovich [PZ12] for such schemes, that we denote by $\mathsf{2Partition}$, and observe that it does not guarantee non-transitive joins.

3.1 Adjustable Join Schemes

An adjustable join scheme for a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, an encoding space $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a column label space $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda\in\mathbb{N}}$, is a 4-tuple $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ of polynomial-time algorithms with the following properties:

- The key-generation algorithm, KeyGen, is a probabilistic algorithm that takes as input a unary representation 1^λ of the security parameter $\lambda \in \mathbb{N}$, and outputs a secret key sk and public parameters params.
- The encoding algorithm, Enc, is a deterministic algorithm that takes as input a secret key sk, a message $m \in \mathcal{M}_\lambda$ and a column label $\mathsf{col} \in \mathcal{L}_\lambda$, and outputs an encoding $c \in \mathcal{C}_\lambda$.
- The token-generation algorithm, $\mathsf{TokenGen}$, is a probabilistic algorithm that takes as input a secret key sk and two column labels $\mathsf{col}_i, \mathsf{col}_j \in \mathcal{L}_\lambda$, and outputs a pair (τ_i, τ_j) of adjustment tokens.
- The adjustment algorithm, Adj, is a deterministic algorithm that takes as input the public parameters params, an encoding $c \in \mathcal{C}_\lambda$ and an adjustment token τ, and outputs an encoding $c' \in \mathcal{C}_\lambda$.

Correctness. In terms of correctness, we require that for all sufficiently large $\lambda \in \mathbb{N}$, and for any two column labels $\mathsf{col}_i, \mathsf{col}_j \in \mathcal{L}_\lambda$ and two messages $m_i, m_j \in \mathcal{M}_\lambda$, it holds that

$$m_i = m_j \Longleftrightarrow \mathsf{Adj}\left(\mathsf{params}, \tau_i, \mathsf{Enc}_{\mathsf{sk}}\left(m_i, \mathsf{col}_i\right)\right) = \mathsf{Adj}\left(\mathsf{params}, \tau_j, \mathsf{Enc}_{\mathsf{sk}}\left(m_j, \mathsf{col}_j\right)\right)$$

with an overwhelming probability over the choice of $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\tau_i, \tau_j) \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}_i, \mathsf{col}_j)$.

A comparison with the notion of Popa and Zeldovich [PZ12]. The above notion of an adjustable join scheme is essentially identical to the one originally formalized by Popa and Zeldovich [PZ12] except for the following minor difference: When computing the join of columns i and j, we allow the scheme to apply the adjustment algorithm to the encodings of column i and to the encodings of column j, whereas Popa and Zeldovich allow the scheme to apply the adjustment algorithm only to the encodings of column j. As far as we can tell, applying the adjustment algorithm to the encodings of both columns is fully compatible with the design of CryptDB.

3.2 The 2Partition Security Notion and its Weakness

The notion of security introduced by Popa and Zeldovich [PZ12], that we denote by $\mathsf{2Partition}$, considers an experiment in which an adversary may adaptively define two disjoint sets of columns, which we refer to as a "left" set L and a "right" set R. The adversary is given the ability to compute joins inside L and joins inside R, but it should not be able to compute the join between any column in L and any column in R. Specifically, at any point in time the adversary can insert any column to either L or R, and to obtain encodings of messages of her choice relative to any of these columns. In addition, the adversary may obtain tokens for computing the join of all columns col_i and col_j such that $\mathsf{col}_i, \mathsf{col}_j \in L$ or $\mathsf{col}_i, \mathsf{col}_j \in R$.

The $\mathsf{2Partition}$ notion of security asks that such an adversary should not be able to compute the join of any two columns $\mathsf{col}_i \in L$ and $\mathsf{col}_j \in R$. This is modeled in the experiment by enabling the adversary to output a pair of messages, m_0 and m_1, and providing the adversary either with the encodings of m_0 for all columns in R or with the encodings of m_1 for all columns in R. The adversary should not be able to distinguish these two cases with a non-negligible advantage (of course, as long as the adversary did not explicitly ask for an encoding of m_0 or m_1 relative to some column label in R).

A comparison with the notion of Popa and Zeldovich [PZ12]. The above informal description is in fact a simplification of the notion considered by Popa and Zeldovich [PZ12], but a straightforward hybrid argument shows that the two are in fact equivalent (whenever the message space is not too small). Specifically, whereas in the above description the adversary obtains *either* encodings of m_0 for all columns in R *or* encodings of m_1 for all columns in R, Popa and Zeldovich provide the adversary with encodings of *both* m_0 and m_1 for all columns in R but in a shuffled order. We refer the reader to [PZ12, Sec. 3.1] for a more formal description of their notion.

2Partition does not capture transitivity. Intuitively, the 2Partition notion guarantees that for any two disjoint sets of columns, L and R, the ability to compute joins inside L and joins inside R, does not imply the ability to compute the join between any column in L and any column in R. However, this does not capture transitivity: Assume that there is a certain column col^* that does not belong to either L or R, then the ability to compute the join between col^* and columns in L, and to compute the join between col^* and columns in R, may imply the ability to compute the join between columns in L and columns in R.

Moreover, it is not only that 2Partition does not capture transitivity, but in fact the adjustable join scheme of Popa and Zeldovich [PZ12] is completely transitive (due to efficiency considerations) although it satisfies 2Partition: For any three columns col_i, col_j and col_k, given tokens for computing the joins between col_i and col_k and between col_k and col_j, it is easy to efficiently construct a token for computing the join between col_i and col_j.

Specifically, as pointed out in Section 1.2, their scheme uses a group $\mathbb{G}$ of prime order p that is generated by an element $g \in \mathbb{G}$, and a pseudorandom function PRF mapping column labels and messages into $\mathbb{Z}_p^*$ with keys $\mathsf{sk_{col}}$ and $\mathsf{sk_{msg}}$, respectively (Popa et al. use a pseudorandom permutation, but in fact any pseudorandom function suffices as any specific collision occurs with only a negligible probability whenever the range of the function is of size super-polynomial in the security parameter). The encoding of a message m for a column col_i is the group element $g^{\mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_i)\cdot\mathsf{PRF}_{\mathsf{sk_{msg}}}(m)} \in \mathbb{G}$, and a token for computing the join between columns col_i and col_j is the element $\tau_{i,j} = \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_i) \cdot \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_j)^{-1} \in \mathbb{Z}_p$. Thus, it is clear that given the tokens $\tau_{i,k} = \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_i) \cdot \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_k)^{-1} \in \mathbb{Z}_p$ and $\tau_{k,j} = \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_k) \cdot \mathsf{PRF}_{\mathsf{sk_{col}}}(\mathsf{col}_j)^{-1} \in \mathbb{Z}_p$, one can efficiently compute the token $\tau_{i,j} = \tau_{i,k} \cdot \tau_{k,j}$.

4 Strengthening the Definitional Framework

In this section we introduce our new and refined framework for capturing the security of adjustable join schemes. First, in Section 4.1, we introduce a new notion of security, denoted 3Partition, which strictly strengthens 2Partition. Our notion considers a partitioning of the columns into *three* disjoint sets (instead of two disjoint sets as in the 2Partition notion) in a manner that enables it to properly model non-transitive joins.

As discussed in Section 1.2, partitioning the columns into three disjoint sets is intuitively inherent for capturing the security of adjustable join schemes. In Sections 4.2 and 4.3 we show that partitioning the columns into three sets is indeed sufficient and captures the security of adjustable join schemes in a natural manner: We formalize natural simulation-based and indistinguishability-based security notion, capturing the “minimal leakage” of join schemes without any explicit partitioning of the columns, and prove that 3Partition in positioned between their adaptive variants and their non-adaptive variant. Finally, in Section 4.4 we include some additional remarks regarding the standard aspects of column privacy and leakage of frequency characteristics that arise in our notions of security.

4.1 The $\mathsf{3Partition}$ Security Notion

Our $\mathsf{3Partition}$ notion of security considers an adversary that may adaptively define three disjoint sets of columns, which we refer to as a "left" set L, a "right" set R, and a "middle" set M. The adversary is given the ability to compute joins inside L, inside M, and inside R, as well as joins between L and M and between R and M, but it should not be able to compute the join between any column in L and any column in R. Specifically, at any point in time the adversary can insert any column to either L, R or M, and to obtain encodings of messages of her choice relative to any of these columns. In addition, the adversary may obtain tokens for computing the join of all columns col_i and col_j such that $\mathsf{col}_i, \mathsf{col}_j \in L \cup M$ or $\mathsf{col}_i, \mathsf{col}_j \in R \cup M$.

The $\mathsf{3Partition}$ notion of security asks that such an adversary should not be able to compute the join of any two columns $\mathsf{col}_i \in L$ and $\mathsf{col}_j \in R$. This is modeled by enabling the adversary to output a pair of messages, m_0 and m_1, and providing the adversary either with the encodings of m_0 for all columns in R or with the encodings of m_1 for all columns in R. The adversary should not be able to distinguish these two cases with a non-negligible advantage (of course, as long as the adversary did not explicitly ask for an encoding of m_0 or m_1 relative to some column label in $R \cup M$).

Definition 4.1. A join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is $\mathsf{3Partition}$-*secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{3Par}}_{\Pi,\mathcal{A}}(\lambda) \overset{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{A}}(\lambda, 1) = 1\right] \right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ the experiment $\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{A}}(\lambda, b)$ is defined as follows:

1. Setup phase: Sample $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and initialize $L = R = M = \emptyset$. The public parameters params are given as input to the adversary $\mathcal{A}$.
2. Pre-challenge query phase: $\mathcal{A}$ may adaptively issue $\mathsf{AddColumn}$, Enc and $\mathsf{TokenGen}$ queries, which are defined as follows.
   - $\mathsf{AddColumn}(\mathsf{col}, S)$: Adds the column label col to the set S, where $S \in \{``L", ``R", ``M"\}$. $\mathcal{A}$ is not allowed to add a column label into more than one set (i.e., the sets L, R and M must always be pairwise disjoint).
   - $\mathsf{Enc}(m, \mathsf{col})$: Computes and returns to $\mathcal{A}$ an encoding $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})$, where $\mathsf{col} \in L \cup R \cup M$.
   - $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$: Computes and returns to $\mathcal{A}$ a pair of tokens $(\tau_i, \tau_j) \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}_i, \mathsf{col}_j)$, where $\mathsf{col}_i, \mathsf{col}_j \in L \cup M$ or $\mathsf{col}_i, \mathsf{col}_j \in R \cup M$.
3. Challenge phase: $\mathcal{A}$ chooses messages m_0 and m_1 subject to the constraint that $\mathcal{A}$ did not previously issue a query of the form $\mathsf{Enc}(m, \mathsf{col})$ where $m \in \{m_0, m_1\}$ and $\mathsf{col} \in R \cup M$. As a response, $\mathcal{A}$ obtains an encoding $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m_b, \mathsf{col})$ for every $\mathsf{col} \in R$.
4. Post-challenge query phase: As in the pre-challenge query phase, with the restriction that $\mathcal{A}$ is not allowed to issue a query of the form $\mathsf{Enc}(m, \mathsf{col})$ where $m \in \{m_0, m_1\}$ and $\mathsf{col} \in R \cup M$. In addition, for each $\mathsf{AddColumn}(\mathsf{col}, ``R")$ query, $\mathcal{A}$ is also provided with $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m_b, \mathsf{col})$.
5. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0,1\}$ which is defined as the value of the experiment.

Our $\mathsf{3Partition}$ notion clearly strengthens the $\mathsf{2Partition}$ notion of Popa and Zeldovich [PZ12] by considering a partitioning of the column labels into three sets instead of two sets. Moreover, as shown in Section 3.2, the adjustable join scheme of Popa and Zeldovich is not a $\mathsf{3Partition}$-secure scheme, although they proved it to be a $\mathsf{2Partition}$-secure scheme, and thus our $\mathsf{3Partition}$ notion strictly strengthens the $\mathsf{2Partition}$ notion.

4.2 Indistinguishability-Based Security Notions

We first introduce some basic notation that will be helpful in formalizing our indistinguishability-based security notions. A *database* DB of dimensions $\mathsf{dim} = \mathsf{dim}(\mathsf{DB}) = (t, (n_i)_{i=1}^t)$ consists of a list of distinct column labels, denoted $\mathsf{Cols} = \mathsf{Cols}(\mathsf{DB}) = (\mathsf{col}_1, \ldots, \mathsf{col}_t)$, and of a list of distinct messages $L_i = (m_1^i, \ldots, m_{n_i}^i)$ for each column label $\mathsf{col}_i \in \mathsf{Cols}$. The size of a database is defined as $\mathsf{size}(\mathsf{DB}) = \sum_{i=1}^t n_i$ (i.e., the total number of messages in DB). We let

$$V = V(\mathsf{dim}) = \{(i,k) | i \in [t], k \in [n_i]\}$$

and view the messages of the database as a map $\mathsf{m} = \mathsf{m}(\mathsf{DB}) : V \to \mathcal{M}_\lambda$ by setting $\mathsf{m}(i,k) = m_k^i$. A map m is "valid" (i.e., can be a part of a description of a database) if and only if $\mathsf{m}(i,k) \neq \mathsf{m}(i,\ell)$ for all $i \in [t]$ and $k \neq \ell \in [n_i]$.

Given an adjustable join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ we extend its encoding algorithm from encoding single messages to encoding a complete database by defining $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}) = \{(i, k, \mathsf{Enc}_{\mathsf{sk}}(\mathsf{m}(i,k), \mathsf{col}_i))\}_{i \in [t], k \in [n_i]}$. Similarly, given a list of pairs of indices $I = ((i_1, j_1), \ldots, (i_s, j_s)) \in ([t] \times [t])^*$, we extend its token-generation algorithm by defining

$$\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}, I) = \{(i, j, \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}_i, \mathsf{col}_j))\}_{(i,j) \in I}.$$

In addition, for such a list I and a database DB we define

$$\mathsf{Join}_{\mathsf{DB}}(i,j) = \{(k,\ell) \in [n_i] \times [n_j] : \mathsf{m}(i,k) = \mathsf{m}(j,\ell)\},$$

and we define the leakage of (DB, I) to be

$$\mathcal{L}(\mathsf{DB}, I) = \Big(\mathsf{dim}(\mathsf{DB}), \mathsf{Cols}(\mathsf{DB}), I, \{(i, j, \mathsf{Join}_{\mathsf{DB}}(i,j))\}_{(i,j) \in I}\Big).$$

Non-adaptive IND security. Our non-adaptive indistinguishability-based notion is perhaps the most simplistic and natural notion: It considers an adversary that obtains the public parameters of the scheme, and then chooses two databases, DB_0 and DB_1, and a list I of pairs of indices such that $\mathcal{L}(\mathsf{DB}_0, I) = \mathcal{L}(\mathsf{DB}_1, I)$ (i.e., the functionality of the scheme does not trivially distinguish DB_0 and DB_1). We ask that such an adversary has only a negligible advantage in distinguishing between $(\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_0), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_0, I))$ and $(\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_1), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_1, I))$. That is, the adversary should essentially not be able to distinguish between an encoding of DB_0 and an encoding of DB_1, where in both cases she is given tokens for computing the joins of all column label pairs corresponding to the pair of indices in I.

Definition 4.2 (Non-adaptive IND security). A join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is *non-adaptively* IND-*secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{naIND}}(\lambda) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{naIND}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{naIND}}(\lambda, 1) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ the experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{naIND}}(\lambda, b)$ is defined as follows:

1. Setup phase: Sample $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The public parameters params are given as input to the adversary $\mathcal{A}$.
2. Challenge phase: $\mathcal{A}$ chooses two databases, DB_0 and DB_1, and a list I of column label pairs such that $\mathcal{L}(\mathsf{DB}_0, I) = \mathcal{L}(\mathsf{DB}_1, I)$. As a response, $\mathcal{A}$ obtains $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_b)$ and $\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_b, I)$.

3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

The following claim, which is proved in Appendix A, states that non-adaptive IND security is implied by $\mathsf{3Partition}$ security.

Claim 4.3. *Any* $\mathsf{3Partition}$*-secure join scheme that supports a message space of super-polynomial size (in the security parameter* $\lambda \in \mathbb{N}$*) is a non-adaptively* IND*-secure join scheme.*

Adaptive IND security. We consider an adaptive flavor of Definition 4.2 by considering adversaries that can adaptively issue encoding queries and token-generation queries. Each encoding query consists of a pair of messages, m_0 and m_1, and a column label col, and the adversary obtains an encoding $\mathsf{Enc}_{\mathsf{sk}}(m_b, \mathsf{col})$ (where $b \in \{0, 1\}$ is fixed throughout the experiment). The adversary's encoding queries define two databases, DB_0 and DB_1, of the same dimension that have the same column label set. Each token-generation query consists of a pair of column labels, and the adversary obtains a token for computing the join of these columns. The adversary's token-generation queries define a set I of all column label pairs for which the adversary has obtained tokens. Such an adversary is called "valid" if at the end of the experiment it holds that $\mathcal{L}(\mathsf{DB}_0, I) = \mathcal{L}(\mathsf{DB}_1, I)$.

Definition 4.4 (Adaptive IND security)**.** A join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is IND-*secure* if for any probabilistic polynomial-time *valid* adversary $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(\lambda, 1) = 1\right] \right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0, 1\}$ the experiment $\mathsf{Exp}^{\mathsf{IND}}_{\Pi,\mathcal{A}}(\lambda, b)$ is defined as follows:

1. Setup phase: Sample $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The public parameters params are given as input to the adversary $\mathcal{A}$.
2. Query phase: $\mathcal{A}$ may adaptively issue Enc and $\mathsf{TokenGen}$ queries, which are defined as follows.
   - $\mathsf{Enc}(m_0, m_1, \mathsf{col})$: Computes an encoding $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m_b, \mathsf{col})$, and returns c to $\mathcal{A}$.
   - $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$: Computes a token $(\tau_i, \tau_j) \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}_i, \mathsf{col}_j)$, and returns (τ_i, τ_j) to $\mathcal{A}$.
3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

The following claim states that adaptive IND security implies $\mathsf{3Partition}$ security.

Claim 4.5. *Any adaptively* IND*-secure join scheme is a* $\mathsf{3Partition}$*-secure join scheme.*

The proof of Claim 4.5 is straightforward, as the IND-security experiment is essentially less restrictive than the $\mathsf{3Partition}$-security experiment. Specifically, given an adversary to the $\mathsf{3Partition}$-security experiment we can construct an adversary to the IND-security experiment (having the exact same advantage) as follows:

- All queries of the form $\mathsf{AddColumn}(\mathsf{col}, S)$ are ignored. However, the adversary keeps track of the set R.
- Any query of the form $\mathsf{Enc}(m, \mathsf{col})$ is converted into a query $\mathsf{Enc}(m, m, \mathsf{col})$.
- Any query of the form $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$ is forwarded as without any modification.
- The challenge (m_0, m_1) is converted into queries of the form $\mathsf{Enc}(m_0, m_1, \mathsf{col})$ for each $\mathsf{col} \in R$.
- Any query of the form $\mathsf{AddColumn}(\mathsf{col}, \text{"}R\text{"})$ in the post-challenge query phase is converted into a query $\mathsf{Enc}(m_0, m_1, \mathsf{col})$.

4.3 Simulation-Based Security Notions

As with our indistinguishability-based notions, we first formalize a non-adaptive simulation-based notion, which we then generalize to an adaptive one.

Non-adaptive SIM security. Our non-adaptive simulation-based notion considers an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$. In the real-world experiment, the adversary $\mathcal{A}$ interacts with the scheme in the following non-adaptive manner: It obtains the public parameters of the scheme, chooses a databases DB and a list I of column label pairs, and then obtains an encoding of DB and tokens for all column label pairs in I. In the ideal-world experiment, the simulator has to produce a view that is indistinguishable from the real world when given only the "minimal" leakage $\mathcal{L}(\mathsf{DB}, I)$, and without being given the database DB (recall that the leakage function $\mathcal{L}$ was defined in Section 4.2).

Formally, for an adjustable join scheme Π and an adversary $\mathcal{A}$, we consider the experiment $\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)$ which is defined as follows:

1. Setup phase: Sample $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The public parameters params are given as input to the adversary $\mathcal{A}$.
2. Challenge phase: $\mathcal{A}$ chooses a databases DB and a list I of column label pairs. As a response, $\mathcal{A}$ obtains $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB})$ and $\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}, I)$.
3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

In addition, given an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, we consider the experiment $\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)$ which is defined as follows:

1. Setup phase: The simulator $\mathcal{S}$ produces the public parameters params, which are given as input to the adversary $\mathcal{A}$.
2. Challenge phase: $\mathcal{A}$ chooses a databases DB and a list I of column label pairs. The simulator is given $\mathcal{L}(\mathsf{DB}, I)$ and produces a database encoding and a list of tokens, which are given to $\mathcal{A}$.
3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

Definition 4.6 (Non-adaptive SIM security). A join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is *non-adaptively* SIM*-secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exist a probabilistic polynomial-time simulator $\mathcal{S}$ and a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{naSIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1\right] \right| \le \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

The following claim, which is proved in Appendix A, states that non-adaptive SIM security is equivalent to non-adaptive IND security.

Claim 4.7. *Any join scheme Π that supports a message space of super-polynomial size (in the security parameter $\lambda \in \mathbb{N}$) is non-adaptively SIM secure if and only if it is non-adaptive IND secure.*

Adaptive SIM security. We consider an adaptive flavor of Definition 4.6 by naturally generalizing the above real-world and ideal-world experiments. Specifically, for an adjustable join scheme Π and an adversary $\mathcal{A}$, we consider the experiment $\mathsf{Real}^{\mathsf{SIM}}_{\Pi,\mathcal{A}}(\lambda)$ which is defined as follows:

1. Setup phase: Sample $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The public parameters params are given as input to the adversary $\mathcal{A}$.

2. Query phase: $\mathcal{A}$ may adaptively issue Enc and $\mathsf{TokenGen}$ queries, which are defined as follows.
   - $\mathsf{Enc}(m, \mathsf{col})$: Computes an encoding $c \leftarrow \mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})$, and returns c to $\mathcal{A}$.
   - $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$: Computes a token $(\tau_i, \tau_j) \leftarrow \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{col}_i, \mathsf{col}_j)$, and returns (τ_i, τ_j) to $\mathcal{A}$.
3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

In addition, given an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$, we consider the experiment $\mathsf{Ideal}^{\mathsf{SIM}}_{\mathcal{A},\mathcal{S}}(\lambda)$ which is defined as follows:

1. Setup phase: The simulator $\mathcal{S}$ produces the public parameters params, which are given as input to the adversary $\mathcal{A}$. An empty database DB and an empty list I of column label pairs are initialized.
2. Query phase: $\mathcal{A}$ may adaptively issue Enc and $\mathsf{TokenGen}$ queries, which are defined as follows.
   - $\mathsf{Enc}(m, \mathsf{col})$: The pair (m, col) is inserted into the database DB, and $\mathcal{S}$ obtains $\mathcal{L}(\mathsf{DB}, I)$. Then, $\mathcal{S}$ provides $\mathcal{A}$ with an encoding c.
   - $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$: The pair $(\mathsf{col}_i, \mathsf{col}_j)$ is inserted into the list I, and $\mathcal{S}$ obtains $\mathcal{L}(\mathsf{DB}, I)$. Then, $\mathcal{S}$ provides $\mathcal{A}$ with a pair (τ_i, τ_j).
3. Output phase: $\mathcal{A}$ outputs a value $\sigma \in \{0, 1\}$ which is defined as the value of the experiment.

Definition 4.8 (Adaptive SIM security)**.** A join scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is SIM-*secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exist a probabilistic polynomial-time simulator $\mathcal{S}$ and a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{SIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda) \overset{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Real}^{\mathsf{SIM}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Ideal}^{\mathsf{SIM}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1\right] \right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

The following claim states that adaptive SIM security implies adaptive IND security.

Claim 4.9. *Any* SIM*-secure join scheme is an* IND*-secure join scheme.*

The proof idea of Claim 4.9 is similar to the non-adaptive case: The adversary cannot distinguish between DB_0 and the simulation, and between the simulation and DB_1, hence cannot distinguish between DB_0 and DB_1. In more details, given an adversary $\mathcal{B}$ to the IND-security experiment, we construct an adversary $\mathcal{A}$ to the SIM-security experiment, which samples $c \leftarrow \{0, 1\}$, and converts each query of the form $\mathsf{Enc}(m_0, m_1, \mathsf{col})$ into a query $\mathsf{Enc}(m_c, \mathsf{col})$. Finally, when $\mathcal{B}$ halts and outputs $\sigma \in \{0, 1\}$ then $\mathcal{A}$ halts and outputs $\sigma \oplus c$. A similar argument to the one in the proof of Claim 4.7 (see Appendix A.4) shows that

$$\mathsf{Adv}^{\mathsf{IND}}_{\Pi,\mathcal{B}}(\lambda) = 2 \cdot \mathsf{Adv}^{\mathsf{SIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda),$$

where $\mathcal{S}$ is the simulator for which $\mathsf{Adv}^{\mathsf{SIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ is negligible. Therefore, the SIM-security of Π implies its IND-security.

4.4 Additional Remarks

Column privacy. Our notions of security include the column labels $\mathsf{Cols}(\mathsf{DB})$ of the encrypted databases as explicit leakage (either as part of the experiment or via leakage functions). In fact, our scheme in Section 5 does not leak the column labels. All of our security notions can be naturally refined to model column privacy in addition to message privacy. Although the task of guaranteeing column privacy is well motivated, in this paper we focus on message privacy in order to simplify our notions of security.

Implicit (and unavoidable) leakage. Our notions of security assume that the given encrypted databases are "valid" in the sense that no message appears more than once in each column. An alternative approach (e.g., [CK10, CGK+11]) is to avoid this assumption, and explicitly include a leakage function that specifies the frequency characteristics of each column. For deterministic encodings of messages (where such leakage is unavoidable), these two approaches are equivalent. Therefore, we do not explicitly include such a leakage function, but rather incorporate this unavoidable leakage directly into our security notions.

5 Our Adjustable Join Scheme

In this section we present an adjustable join scheme that satisfies our $\mathsf{3Partition}$ security notion. In Section 5.1 we describe our scheme and prove its correctness, and in Section 5.2 we prove its security.

5.1 The Scheme

Let $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ be a pseudorandom function family, and let $\mathcal{G}$ be a probabilistic polynomial-time algorithm that takes as input the security parameter 1^λ, and outputs a triplet $(\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, where p is a λ-bit prime number, $\mathbb{G}$ and $\mathbb{G}_T$ are groups of order p, g is a generator of $\mathbb{G}$, and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate efficiently-computable bilinear map. The scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{TokenGen}, \mathsf{Adj})$ is defined as follows.

- **Key generation.** On input 1^λ the key-generation algorithm KeyGen samples $(\mathbb{G}, \mathbb{G}_T, g, p, \hat{e}) \leftarrow \mathcal{G}(1^\lambda)$, $K_1 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, and $K_2 \leftarrow \mathsf{PRF.Gen}(1^\lambda)$. For each $i \in \{1, 2\}$ we let $\mathsf{PRF}_{K_i}(\cdot) = \mathsf{PRF.Eval}(K_i, \cdot)$, and we assume that $\mathsf{PRF}_{K_1} : \mathcal{M}_\lambda \to \mathbb{Z}_p^4$ and $\mathsf{PRF}_{K_2} : \mathcal{L}_\lambda \to \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$, where $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{L} = \{\mathcal{L}_\lambda\}_{\lambda\in\mathbb{N}}$ are the message space and the column label space, respectively. The algorithm outputs $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$ and $\mathsf{sk} = (K_1, K_2)$.

  For the above description, recall that $\mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ denotes the set of all invertible 4×4 matrices over $\mathbb{Z}_p$. Note that a pseudorandom function $\mathsf{PRF}_{K_2} : \mathcal{M}_\lambda \to \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ can be constructed, for example, by taking any pseudorandom function $\mathsf{PRF}_{K_2} : \mathcal{M}_\lambda \to \mathbb{Z}_p^{4\times 4}$ and substituting each non-invertible output with the identity matrix.

- **Encoding.** On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, $\mathsf{sk} = (K_1, K_2)$, a column label $\mathsf{col} \in \mathcal{L}_\lambda$ and a message $m \in \mathcal{M}_\lambda$, the encoding algorithm Enc computes $A_{\mathsf{col}} = \mathsf{PRF}_{K_2}(\mathsf{col}) \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and $x_m = \mathsf{PRF}_{K_1}(m) \in \mathbb{Z}_p^4$, and then outputs $c = g^{A_{\mathsf{col}} x_m} \in \mathbb{G}^4$.

- **Token generation.** On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, $\mathsf{sk} = (K_1, K_2)$ and two column labels $\mathsf{col}, \mathsf{col}' \in \mathcal{L}_\lambda$, the token-generation algorithm $\mathsf{TokenGen}$ uniformly samples $v \leftarrow \mathbb{Z}_p^4 \setminus \{(0,0,0,0)\}$, computes $A_{\mathsf{col}} = \mathsf{PRF}_{K_2}(\mathsf{col}) \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and $A_{\mathsf{col}'} = \mathsf{PRF}_{K_2}(\mathsf{col}') \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$, and then outputs the pair of adjustment tokens $\left(g^{v^\mathsf{T} A_{\mathsf{col}}^{-1}}, g^{v^\mathsf{T} A_{\mathsf{col}'}^{-1}}\right) \in \mathbb{G}^4 \times \mathbb{G}^4$.

- **Adjustment.** On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, an adjustment token $\tau \in \mathbb{G}^4$, and an encoding $c \in \mathbb{G}^4$, the adjustment algorithm Adj outputs $\hat{e}(\tau^\mathsf{T}, c) \in \mathbb{G}_T$.

Correctness. For any two column labels $\mathsf{col}, \mathsf{col}' \in \mathcal{L}_\lambda$ and for any two messages $m, m' \in \mathcal{M}_\lambda$ it holds that

$$\begin{aligned}\mathsf{Adj}\left(\tau, \mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})\right) &= \hat{e}\left(g^{v^\mathsf{T} A_{\mathsf{col}}^{-1}}, g^{A_{\mathsf{col}} x_m}\right)\\ &= \hat{e}(g,g)^{v^\mathsf{T} A_{\mathsf{col}}^{-1} A_{\mathsf{col}} x_m} = \hat{e}(g,g)^{v^\mathsf{T} x_m}\\ \mathsf{Adj}\left(\tau', \mathsf{Enc}_{\mathsf{sk}}(m', \mathsf{col}')\right) &= \hat{e}\left(g^{v^\mathsf{T} A_{\mathsf{col}'}^{-1}}, g^{A_{\mathsf{col}'} x_{m'}}\right)\\ &= \hat{e}(g,g)^{v^\mathsf{T} A_{\mathsf{col}'}^{-1} A_{\mathsf{col}'} x_{m'}} = \hat{e}(g,g)^{v^\mathsf{T} x_{m'}},\end{aligned}$$

where $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $(\tau, \tau') \leftarrow \mathsf{Adj}(\mathsf{sk}, \mathsf{col}, \mathsf{col}')$. Therefore, it holds that

$$\mathsf{Adj}\left(\tau, \mathsf{Enc}_{\mathsf{sk}}(m, \mathsf{col})\right) = \mathsf{Adj}\left(\tau', \mathsf{Enc}_{\mathsf{sk}}(m', \mathsf{col}')\right)$$

if and only if $v^\mathsf{T} x_m = v^\mathsf{T} x_{m'}$. Note that if $m = m'$ then the equality always holds. In addition, if $m \neq m'$ then with an overwhelming probability $x_m \neq x_{m'}$ (since PRF is a pseudorandom function), and since v is uniform then the probability that $v^\mathsf{T} x_m = v^\mathsf{T} x_{m'}$ is at most $1/p$. We conclude that if $m \neq m'$ then $v^\mathsf{T} x_m \neq v^\mathsf{T} x_{m'}$ with an overwhelming probability.

5.2 Proof of Security

We prove the following theorem:

Theorem 5.1. *Assuming that* PRF *is a pseudorandom function family and that the linear assumption holds relative to* $\mathcal{G}$*, then* Π *is a* $\mathsf{3Partition}$*-secure adjustable join scheme.*

For proving Theorem 5.1, we introduce a scheme $\hat{\Pi}$ which is obtained from Π by replacing its $\mathsf{TokenGen}$ and Adj algorithms with the followings algorithms:

- **Token generation.** On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, $\mathsf{sk} = (K_1, K_2)$ and two column labels $\mathsf{col}, \mathsf{col}' \in \mathcal{L}_\lambda$, the modified token-generation algorithm $\mathsf{TokenGen}$ uniformly samples $V \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$, computes $A_{\mathsf{col}} = \mathsf{PRF}_{K_2}(\mathsf{col}) \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$ and $A_{\mathsf{col}'} = \mathsf{PRF}_{K_2}(\mathsf{col}') \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$, and then outputs $\left(g^{VA_{\mathsf{col}}^{-1}}, g^{VA_{\mathsf{col}'}^{-1}}\right) \in \mathbb{G}^{4\times4} \times \mathbb{G}^{4\times4}$.
- **Adjustment.** On input $\mathsf{params} = (\mathbb{G}, \mathbb{G}_T, g, p, \hat{e})$, an adjustment token $\mathcal{T} \in \mathbb{G}^{4\times4}$, and an encoding $c \in \mathbb{G}^4$, the modified adjustment algorithm Adj outputs $\hat{e}(\mathcal{T}, c) \in \mathbb{G}_T^4$.

Note that Π can be obtained from $\hat{\Pi}$ by viewing any $v \in \mathbb{Z}_p^4 \setminus \{(0,0,0,0)\}$ that is produced by Π's token-generation algorithm as the first row of the matrix $V \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$ that is produced by $\hat{\Pi}$ token-generation algorithm. That is, Π's token-generation algorithm can be obtained from $\hat{\Pi}$'s token-generation algorithm by outputting only the first rows of its tokens. Thus, there is no information that Π reveals and $\hat{\Pi}$ does not, and therefore it suffices to prove the security of $\hat{\Pi}$.

For each $b \in \{0,1\}$ and an adversary $\mathcal{A}$, let $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, b)$ denote the experiment obtained from $\mathsf{Exp}^{\mathsf{3Par}}_{\hat{\Pi},\mathcal{A}}(\lambda, b)$ by replacing the pseudorandom functions $\mathsf{PRF}_{K_1} : \mathcal{M}_\lambda \to \mathbb{Z}_p^4$ and $\mathsf{PRF}_{K_2} : \mathcal{L}_\lambda \to \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$ with truly random functions $f_1 : \mathcal{M}_\lambda \to \mathbb{Z}_p^4$ and $f_2 : \mathcal{L}_\lambda \to \mathsf{Rk}_4(\mathbb{Z}_p^{4\times4})$. By the pseudorandomness property of the pseudorandom function family PRF, it holds that for any $b \in \{0,1\}$ and any probabilistic polynomial-time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in distinguishing between the experiments $\mathsf{Exp}^{\mathsf{3Par}}_{\hat{\Pi},\mathcal{A}}(\lambda, b)$ and $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, b)$ is negligible. Therefore, in order to prove the $\mathsf{3Partition}$-security of $\hat{\Pi}$ it suffices to show that the advantage of any adversary $\mathcal{A}$ in distinguishing between the experiments $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 1)$ is negligible.

By Claim 2.5 and Fact 2.3, it follows that under the linear assumption it holds that

$$\begin{aligned}\left(\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{Bx}\right) &\approx_c \left(\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{Bz}\right)\\ &\approx_c \left(\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{By}\right),\end{aligned}$$

where $\mathsf{params} \leftarrow \mathcal{G}(1^\lambda)$, $A, B \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and $x, y, z \leftarrow \mathbb{Z}_p^4$. We denote by X and Y the computationally indistinguishable ensembles $X = (\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{Bx})$ and $Y = (\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{By})$. Assume for now that during the pre-challenge query phase, the adversary $\mathcal{A}$ does not issue a query of the form $\mathsf{Enc}(m_0, \mathsf{col})$ or $\mathsf{Enc}(m_1, \mathsf{col})$, from any column label col, where m_0 and m_1 are the challenge messages. We claim that there exists a polynomial-time challenger Chal, such that it holds that $\mathsf{Chal}^{\mathcal{A}}(X) \equiv \mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0)$ and $\mathsf{Chal}^{\mathcal{A}}(Y) \equiv \mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0)$ as distributions (and this implies that the advantage of $\mathcal{A}$ in distinguishing between the experiments $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0)$ and $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 1)$ is negligible subject to the above assumption on $\mathcal{A}$). Given $(\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{Bz})$ as input and $\mathcal{A}$ as oracle, the challenger Chal works as follows:

Setup phase. Chal provides $\mathcal{A}$ with params.

Pre-challenge query phase. We specify how Chal handles $\mathcal{A}$'s queries:

- $\mathsf{AddColumn}(\mathsf{col}, S)$: Chal adds the column label col to the set S, where $S \in \{\text{“}L\text{”}, \text{“}R\text{”}, \text{“}M\text{”}\}$. In addition, Chal samples $R_{\mathsf{col}} \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$, and denotes

$$A_{\mathsf{col}} = \begin{cases} R_{\mathsf{col}}A & \mathsf{col} \in L \\ R_{\mathsf{col}}B & \mathsf{col} \in R \\ R_{\mathsf{col}} & \mathsf{col} \in M \end{cases}.$$

  Note that since Chal does not explicitly know A and B, he does not explicitly know A_{col} in case that $\mathsf{col} \in L \cup R$.

- $\mathsf{Enc}(m, \mathsf{col})$: Chal samples $x_m \leftarrow \mathbb{Z}_p^4$, unless it was already sampled before. Then, Chal returns $c = g^{A_{\mathsf{col}}x_m}$ to $\mathcal{A}$. We need to show that Chal can efficiently compute c, and we show this by cases:
  1. $\mathsf{col} \in M$: Chal explicitly knows $A_{\mathsf{col}} = R_{\mathsf{col}}$ and x_m, so he can efficiently compute $g^{A_{\mathsf{col}}x_m}$.
  2. $\mathsf{col} \in L$: Since Chal knows g^A, R_{col} and x_m, he can efficiently compute $g^{A_{\mathsf{col}}x_m} = {}^{R_{\mathsf{col}}}\left(g^A\right)^{x_m}$.
  3. $\mathsf{col} \in R$: Similar to the previous case, but with g^B.

- $\mathsf{TokenGen}(\mathsf{col}_i, \mathsf{col}_j)$: Chal returns to $\mathcal{A}$ the pair of tokens $\left(g^{VA^{-1}_{\mathsf{col}_i}}, g^{VA^{-1}_{\mathsf{col}_j}}\right)$ where $V \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ is freshly sampled. We show that Chal is able to efficiently compute τ by cases:
  1. $\mathsf{col}_i, \mathsf{col}_j \in M$: Since Chal explicitly knows $A_{\mathsf{col}_i} = R_{\mathsf{col}_i}$ and $A_{\mathsf{col}_j} = R_{\mathsf{col}_j}$, he can simply sample $V \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$, and compute $g^{VA^{-1}_{\mathsf{col}_i}}$ and $g^{VA^{-1}_{\mathsf{col}_j}}$.
  2. $\mathsf{col}_i, \mathsf{col}_j \in L$: Denote $U = VA^{-1}_{\mathsf{col}_i}$ and $W = VA^{-1}_{\mathsf{col}_j}$. Chal needs to be able to compute g^U and g^W. Fixing A_{col_i} and A_{col_j}, both U and W are uniform in $\mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$, but dependent of each other by the relation $UA_{\mathsf{col}_i} = WA_{\mathsf{col}_j}$. In our case, $A_{\mathsf{col}_i} = R_{\mathsf{col}_i}A$ and $A_{\mathsf{col}_j} = R_{\mathsf{col}_j}A$, so the relation turns into $UR_{\mathsf{col}_i} = WR_{\mathsf{col}_j}$, and Chal can sample $U \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and take $W = UR_{\mathsf{col}_i}R^{-1}_{\mathsf{col}_j}$. Since Chal explicitly knows U and W, he can compute g^U and g^W efficiently.
  3. $\mathsf{col}_i, \mathsf{col}_j \in R$: Similar to the previous case.

4. $\mathsf{col}_i \in L$ and $\mathsf{col}_j \in M$: In this case, $A_{\mathsf{col}_i} = R_{\mathsf{col}_i} A$ and $A_{\mathsf{col}_j} = R_{\mathsf{col}_j}$, so the relation $UA_{\mathsf{col}_i} = WA_{\mathsf{col}_j}$ turns into $UR_{\mathsf{col}_i} A = WR_{\mathsf{col}_j}$. So Chal can sample $U \leftarrow \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ and take $W = UR_{\mathsf{col}_i} AR_{\mathsf{col}_j}^{-1}$. Since Chal explicitly knows U, he can compute g^U. Since he knows U, R_{col_i}, R_{col_j} and g^A, he can efficiently compute $g^W = {}^{UR_{\mathsf{col}_i}}\left(g^A\right)^{R_{\mathsf{col}_j}^{-1}}$.
5. $\mathsf{col}_i \in R$ and $\mathsf{col}_j \in M$: Similar to the previous case.
6. $\mathsf{col}_i \in L$ and $\mathsf{col}_j \in R$: This case is not allowed by the definition of $\mathsf{3Partition}$-security.

Challenge phase. $\mathcal{A}$ chooses messages m_0 and m_1. As a response, Chal returns to $\mathcal{A}$ an encoding $c = g^{A_{\mathsf{col}} z}$ for every $\mathsf{col} \in R$. Since $c = {}^{R_{\mathsf{col}}}\left(g^{Bz}\right)$, and Chal knows R_{col} and g^{Bz}, it can efficiently compute c.

Post-challenge query phase. The only differences from the pre-challenge query phase are the followings:

- $\mathsf{AddColumn}(\mathsf{col}, S)$: In case that $S =$ "R", Chal provides $\mathcal{A}$ with $c = g^{A_{\mathsf{col}} z}$, which we already saw that Chal can efficiently compute.
- $\mathsf{Enc}(m, \mathsf{col})$: In case that $m = m_0$ or $m = m_1$, by the definition of $\mathsf{3Partition}$-security it must be that $\mathsf{col} \in L$, and Chal return to $\mathcal{A}$ the encoding $g^{A_{\mathsf{col}} x}$ or $g^{A_{\mathsf{col}} y}$, respectively. Since $g^{A_{\mathsf{col}} x} = {}^{R_{\mathsf{col}}}\left(g^{Ax}\right)$ and $g^{A_{\mathsf{col}} y} = {}^{R_{\mathsf{col}}}\left(g^{Ay}\right)$, Chal can efficiently compute them.

Output phase. Chal outputs the value $\sigma \in \{0, 1\}$ that $\mathcal{A}$ outputs.

This completes the description of Chal. Denote $x_{m_0} = x$ and $x_{m_1} = y$. It does not cause ambiguity in the notation because we assume that $\mathcal{A}$ does not query m_0 or m_1 in the pre-challenge query phase, so Chal never samples x_{m_0} and x_{m_1} by himself. Every $x_m \in \mathbb{Z}_p^4$ and $A_{\mathsf{col}} \in \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})$ are uniformly random. So Chal returns to $\mathcal{A}$ encodings and tokens with respect to truly random functions. In the case that Chal is given as input $X = (\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{Bx})$, it answers the challenge with encodings of m_0, so we obtain the experiment $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0)$. Similarly, in the case Chal is given $Y = (\mathsf{params}, g^A, g^{Ax}, g^{Ay}, g^B, g^{By})$, we obtain the experiment $\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 1)$. This completes the proof of security for adversaries that fulfill the aforementioned assumption.

When dealing with adversaries that may query m_0 and m_1 in the pre-challenge phase, the problem is that Chal does not know when he queried on m_0 and m_1. If he knew that, then he could respond in the same way he does in the post-challenge query phase. So to solve this, Chal guesses when it is queried with m_0 or m_1. More precisely, let $q(\lambda)$ be a bound on the number of queries that $\mathcal{A}$ performs. Chal samples $t_0, t_1 \leftarrow \{0, \ldots, q(\lambda)\}$. During the pre-challenge phase, if Chal is queried for an encoding of a message m that is the t_0-th or t_1-th distinct message so far, then he acts as if it was queried on m_0 or m_1 respectively, that is, he returns to $\mathcal{A}$ the encoding $g^{A_{\mathsf{col}} x}$ or $g^{A_{\mathsf{col}} y}$, respectively. Then, in the challenge phase, if it turns out that the guess was wrong, or if Chal was queried on less than $\max\{t_0, t_1\}$ distinct messages, then Chal aborts and outputs 0. Since until the challenge phase, the view of $\mathcal{A}$ is independent of the sampling of t_0 and t_1, it holds that the guess of Chal succeeds with probability of exactly $1/(q(\lambda) + 1)^2$, and that the success probability is independent of the behavior of $\mathcal{A}$, so it holds that,

$$\left|\Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\mathsf{Rand},\mathcal{A}}(\lambda, 1) = 1\right]\right| \tag{5.1}$$

$$= (q(\lambda) + 1)^2 \cdot \left|\Pr\left[\mathsf{Chal}^{\mathcal{A}}(X) = 1\right] - \Pr\left[\mathsf{Chal}^{\mathcal{A}}(Y) = 1\right]\right|. \tag{5.2}$$

For any probabilistic polynomial-time adversary $\mathcal{A}$, the bound $q(\lambda)$ on its number of queries is polynomial in the security parameter λ. The linear assumption implies that the expression in Equation (5.2) is negligible, and therefore also the expression in Equation (5.1) is negligible, and this concludes the proof.

Acknowledgments

We thank Zvika Brakerski for fruitful discussions and the TCC reviewers for their valuable comments.

References

[AFG$^+$06] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, 2006.

[ANS$^+$16] G. Asharov, M. Naor, G. Segev, and I. Shahaf. Searchable symmetric encryption: Optimal locality in linear space via two-dimensional balanced allocations. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 1101–1114, 2016.

[BBS98] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology – EUROCRYPT '98*, pages 127–144, 1998.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO '04*, pages 41–55, 2004.

[BHH$^+$08] D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In *Advances in Cryptology – CRYPTO '08*, pages 108–125, 2008.

[BHJ$^+$14] C. Bösch, P. H. Hartel, W. Jonker, and A. Peter. A survey of provably secure searchable encryption. *ACM computing surveys*, 47(2):1–18, 2014.

[BKS16] Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *Advances in Cryptology – EUROCRYPT '16*, pages 852–880, 2016.

[BLR$^+$15] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In *Advances in Cryptology – EUROCRYPT '15*, pages 563–594, 2015.

[CGK$^+$06] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 79–88, 2006.

[CGK$^+$11] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[CGP$^+$15] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, pages 668–679, 2015.

[CJJ$^+$13] D. Cash, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in Cryptology - CRYPTO '13*, pages 353–373, 2013.

[CJJ+14] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014.

[CK10] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *Advances in Cryptology - ASIACRYPT '10*, pages 577–594, 2010.

[CM05] Y.-C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Proceedings of the 3rd International Conference on Applied Cryptography and Network Security*, pages 442–455, 2005.

[CT14] D. Cash and S. Tessaro. The locality of searchable symmetric encryption. In *Advances in Cryptology - EUROCRYPT '14*, pages 351–368, 2014.

[EHK+17] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, 2017.

[FHN+16] M. J. Freedman, C. Hazay, K. Nissim, and B. Pinkas. Efficient set intersection with simulation-based security. *Journal of Cryptology*, 29(1):115–155, 2016.

[FI12] J. Furukawa and T. Isshiki. Controlled joining on encrypted relational database. In *Proceedings of the 5th International Conference on Pairing-Based Cryptography*, pages 46–64, 2012.

[FVY+17] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham. SoK: Cryptographically protected database search. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, pages 172–191, 2017.

[GGG+14] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Advances in Cryptology – EUROCRYPT '14*, pages 578–602, 2014. Merge of [GGJ+13] and [GKL+13].

[GGJ+13] S. Goldwasser, V. Goyal, A. Jain, and A. Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013.

[GKL+13] S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013.

[Goh03] E. Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003.

[GSB+16] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-abuse attacks against order-revealing encryption. Cryptology ePrint Archive, Report 2016/895, 2016.

[HEK12] Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, 2012.

[HIL+02] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 216–227, 2002.

[HIM04] H. Hacigümüs, B. R. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Proceedings of the 9th International Conference on Database Systems for Advances Applications*, pages 125–136, 2004.

[HKD15] I. Hang, F. Kerschbaum, and E. Damiani. ENKI: Access control for encrypted query processing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 183–196, 2015.

[ID03] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, 2003.

[KKN$^+$16] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1329–1340, 2016.

[KM16] S. Kamara and T. Moataz. SQL on structurally-encrypted databases. Cryptology ePrint Archive, Report 2016/453, 2016.

[KO12] K. Kurosawa and Y. Ohtaki. UC-secure searchable symmetric encryption. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, pages 285–298, 2012.

[KO13] K. Kurosawa and Y. Ohtaki. How to update documents verifiably in searchable symmetric encryption. In *Proceedings of the 12th International Conference on Cryptology and Network Security*, pages 309–328, 2013.

[KP13] S. Kamara and C. Papamanthou. Parallel and dynamic searchable symmetric encryption. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, pages 258–274, 2013.

[KPR12] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 19th ACM Conference on Computer and Communications Security*, pages 965–976, 2012.

[NKW15] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 644–655, 2015.

[NS12] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. *SIAM Journal on Computing*, 41(4):772–814, 2012.

[Pop14] R. A. Popa. Building Practical Systems that Compute on Encrypted Data. PhD thesis, Massachusetts Institute of Technology, 2014. Available at `http://www.eecs.berkeley.edu/~raluca/Thesis.pdf`.

[PRZ$^+$11] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, pages 85–100, 2011.

[PRZ$^+$12] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Processing queries on an encrypted database. *Communications of the ACM*, 55(9):103–111, 2012.

[PSS+15] B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *Proceedings of the 24th USENIX Security Symposium*, pages 515–530, 2015.

[PZ12] R. A. Popa and N. Zeldovich. Cryptographic treatment of CryptDB's adjustable join. Technical Report MIT-CSAIL-TR-2012-006, 2012. Available at `http://people.csail.mit.edu/nickolai/papers/popa-join-tr.pdf`.

[PZB15] R. A. Popa, N. Zeldovich, and H. Balakrishnan. Guidelines for using the CryptDB system securely. Cryptology ePrint Archive, Report 2015/979, 2015.

[SWP00] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proceedings of the 21st Annual IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[vLSD+10] P. van Liesdonk, S. Sedghi, J. Doumen, P. H. Hartel, and W. Jonker. Computationally efficient searchable symmetric encryption. In *Proceedings of 7th VLDB Workshop on Secure Data Management*, pages 87–100, 2010.

A Additional Proofs

A.1 Proof of Claim 2.5

The matrix linear assumption. The matrix linear assumption [BHH+08, NS12] states that for all $2 \leq a \leq b$ it holds that

$$\left(\mathsf{params}, g^{R}\right) \approx_c \left(\mathsf{params}, g^{S}\right),$$

where $\mathsf{params} \leftarrow \mathcal{G}(1^\lambda)$, $S \leftarrow \mathbb{Z}_p^{a\times b}$ and $R \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{a\times b})$.

Claim A.1 ([BHH+08, NS12])**.** *The linear assumption implies the matrix linear assumption.*

Claim A.2. *The matrix linear assumption implies that for any $4 \leq a \leq b$ it holds that*

$$\left(\mathsf{params}, g^{A}, g^{Ax}, g^{B}, g^{By}\right) \approx_c \left(\mathsf{params}, g^{A}, g^{Ax}, g^{B}, g^{Bx}\right),$$

where $\mathsf{params} \leftarrow \mathcal{G}(1^\lambda)$, $A, B \leftarrow \mathbb{Z}_p^{a\times b}$ and $x, y \leftarrow \mathbb{Z}_p^{a}$.

We consider the case where $a = b = 4$. For $S \leftarrow \mathbb{Z}_p^{4\times 4}$ it holds that $\Pr[S \notin \mathsf{Rk}_4(\mathbb{Z}_p^{4\times 4})] \leq 2/p$, which is negligible in λ. Thus, from Claim A.1, Claim A.2 and Fact 2.2 we deduce Claim 2.5.

Proof of Claim A.2. Let $\mathsf{params} \leftarrow \mathcal{G}(1^\lambda)$, $A, B \leftarrow \mathbb{Z}_p^{a\times b}$, $A', B' \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{a\times b})$ and $x, y \leftarrow \mathbb{Z}_p^{a}$. We claim that

$$\begin{aligned}
\left(\mathsf{params}, g^{A}, g^{Ax}, g^{B}, g^{By}\right) &\approx_c \left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B}, g^{By}\right) && \text{(A.1)}\\
&\approx_c \left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B'}, g^{B'y}\right) && \text{(A.2)}\\
&\approx_s \left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B'}, g^{B'x}\right) && \text{(A.3)}\\
&\approx_c \left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B}, g^{Bx}\right) && \text{(A.4)}\\
&\approx_c \left(\mathsf{params}, g^{A}, g^{Ax}, g^{B}, g^{Bx}\right), && \text{(A.5)}
\end{aligned}$$

and the claim is deduced by the transitivity of computational indistinguishability. Equations (A.1), (A.2), (A.4) and (A.5) follow directly from the matrix linear assumption and Fact 2.3. It remains to justify Equation (A.3). First, we show that

$$\begin{pmatrix} \mathsf{params} & a_1 & a_2 & \langle a_1, x\rangle & \langle a_1, x\rangle \\ & b_1 & b_2 & \langle b_1, y\rangle & \langle b_1, y\rangle \end{pmatrix} \approx_s \begin{pmatrix} \mathsf{params} & a_1 & a_2 & \langle a_1, x\rangle & \langle a_1, x\rangle \\ & b_1 & b_2 & \langle b_1, x\rangle & \langle b_1, x\rangle \end{pmatrix}, \tag{A.6}$$

where $a_1, a_2, x, y \leftarrow \mathbb{Z}_p^b$, (a_1, a_2) are conditioned to be linearly independent, and also (b_1, b_2) are conditioned so. The probability that (a_1, a_2, b_1, b_2) are linearly dependent is negligible, so by Fact 2.2 we can condition them to be linearly independent. Fixing (a_1, a_2, b_1, b_2) that are linearly independent, $(\langle a_1, x\rangle, \langle a_2, x\rangle, \langle b_1, x\rangle, \langle b_2, x\rangle)$ is uniform over $\mathbb{Z}_p^4$, and clearly this is still true when replacing some of the x's with y's, so we get the same distribution (after the conditioning). Now, we can sample $A' \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{a\times b})$ by sampling $S \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{a\times 2})$ and $T \leftarrow \mathsf{Rk}_2(\mathbb{Z}_p^{2\times b})$ and taking $A' = ST$. We may take the rows of T to be (a_1, a_2). To compute $A'x$ (without knowing x) we have $STx = S[\langle a_1, x\rangle, \langle a_2, x\rangle]^{\mathsf{T}}$ and similarly we can compute $B'x$ and $B'y$, so we can apply a random function to both sides of Equation (A.6) and by Fact 2.1 we get

$$\left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B'}, g^{B'y}\right) \approx_s \left(\mathsf{params}, g^{A'}, g^{A'x}, g^{B'}, g^{B'x}\right).$$

■

A.2 Tools for Proving Claims 4.3 and 4.7

In this section we state and prove Lemma A.5 which will be used to prove Claim 4.3, and we state and prove Lemma A.3 which will be used to prove Lemma A.5 and Claim 4.7. Recall the definitions of a database and a message map from Section 4.2.

Lemma A.3. *There exists a deterministic polynomial-time algorithm* ConstructDB*, that given a security parameter* 1^λ *and a leakage output* $\mathcal{L} = \mathcal{L}(\mathsf{DB}, I)$ *as input, under the assumption that* $|\mathcal{M}_\lambda| \geq \mathsf{size}(\mathsf{DB})$*, outputs a database* DB^* *for which it holds that* $\mathcal{L}(\mathsf{DB}, I) = \mathcal{L}(\mathsf{DB}^*, I)$*. Moreover, let* $\mathsf{dim}(\mathsf{DB}) = (t, (n_i)_{i=1}^t)$*, if* m *is the message map of* DB*, and* m^* *is the message map of* DB^**, then for all* $i, j \in [t]$*,* $k \in [n_i]$ *and* $\ell \in [n_j]$ *it holds that*

$$\mathsf{m}^*(i, k) = \mathsf{m}^*(j, \ell) \quad \Longrightarrow \quad \mathsf{m}(i, k) = \mathsf{m}(j, \ell).$$

Proof. Let

$$\mathcal{L} = (\mathsf{dim} = (t, (n_i)_{i=1}^t), \mathsf{Cols}, I = (i_a, j_a)_{a=1}^s, (J_a)_{a=1}^s)$$

be a possible output of the leakage function. For a valid message map $\mathsf{m}' : V(\mathsf{dim}) \to \mathcal{M}_\lambda$, we denote by $\mathsf{DB}(\mathsf{m}')$ a database with dimensions dim, columns Cols, and message map m'. Clearly $\mathcal{L}(\mathsf{DB}(\mathsf{m}')) = \mathcal{L}$ if and only if for each $a \in [s]$, $k \in [n_{i_a}]$ and $\ell \in [n_{j_a}]$ it holds that

$$\mathsf{m}'(i_a, k) = \mathsf{m}'(j_a, \ell) \quad \Longleftrightarrow \quad (k, \ell) \in J_a,$$

and in that case we say that m' *conforms* with $\mathcal{L}$. Now, we define a graph $G(\mathcal{L})$, whose vertex set is $V(\mathsf{dim})$, and for each $(i_a, j_a) \in I$ we add the edges $\{\{(i_a, k), (j_a, \ell)\} | (k, \ell) \in J_a\}$. Our construction algorithm gets as input $\mathcal{L} = \mathcal{L}(\mathsf{DB}, I)$, builds $G = G(\mathcal{L})$, finds the connected components $(V_1, \ldots, V_c)$ in G, selects an arbitrary injective function $\pi : [c] \to \mathcal{M}_\lambda$,[3] and outputs $\mathsf{DB}^* = \mathsf{DB}(\mathsf{m}^*)$ where

$$\mathsf{m}^*(i, k) = \pi(j) \quad \Longleftrightarrow \quad (i, k) \in V_j.$$

[3]This can be done under the assumption that $\mathcal{M}_\lambda$ is efficiently enumerable, i.e., there exists a linear ordering of $\mathcal{M}_\lambda$ such that we can efficiently compute the first element of $\mathcal{M}_\lambda$, and given $m \in \mathcal{M}_\lambda$ we can efficiently compute the next element after m. This is the case, for example, when $\mathcal{M}_\lambda$ is the set of all bit strings of a certain length.

It remains to show that m^* is valid and conforms with $\mathcal{L}$. To achieve that, we first claim that if $\mathsf{m}^*(i,k) = \mathsf{m}^*(j,\ell)$ then $\mathsf{m}(i,k) = \mathsf{m}(j,\ell)$ where m is the message map of the original DB. Indeed equality in m^* means that there is a path between (i,k) and (j,ℓ) in G, but every consecutive vertices in the path must have the same value of m because m conforms with $\mathcal{L}$, so by the transitivity of equality we get $\mathsf{m}(i,k) = \mathsf{m}(j,\ell)$ as claimed. As a direct consequence we get that m^* is also valid, i.e. it has distinct messages in each column, and that

$$\mathsf{m}^*(i_a,k) = \mathsf{m}^*(j_a,\ell) \quad \Longrightarrow \quad \mathsf{m}(i_a,k) = \mathsf{m}(j_a,\ell) \quad \Longrightarrow \quad (k,\ell) \in J_a$$

The direction $(k,\ell) \in J_a \Rightarrow m^*(i_a,k) = m^*(j_a,\ell)$ is easy since (i_a,k) and (j_a,ℓ) are neighbors in G, hence are in the same connected component. ■

We say that two message maps $\mathsf{m}, \mathsf{m}' : V \to \mathcal{M}_\lambda$ are *equivalent* if $\mathsf{m}' = \pi \circ \mathsf{m}$ for some injective function $\pi : \mathsf{m}(V) \to \mathcal{M}_\lambda$.

Claim A.4. *Let* DB *be a database with dimensions* $\mathsf{dim} = (t, (n_i)_{i=1}^t)$ *and a message map* $\mathsf{m} : V \to \mathcal{M}_\lambda$*, and let* I *be a list of pair of indices. Then, under the assumption that* $|\mathcal{M}_\lambda| \geq \mathsf{size}(\mathsf{DB})$*, we can efficiently construct a sequence of message maps* $\mathsf{m}_0, \ldots, \mathsf{m}_r$ *with the following properties:*

- $r \leq \mathsf{size}(\mathsf{DB})$.
- *For all* $0 \leq q \leq r$, $\mathsf{m}_q : V \to \mathcal{M}_\lambda$ *is valid and conforms with* $\mathcal{L}(\mathsf{DB}, I)$.
- $\mathsf{m}_0 = \mathsf{m}$ *and* m_r *is equivalent to* m^**, where* m^* *is the message map of* $\mathsf{ConstructDB}(1^\lambda, \mathcal{L}(\mathsf{DB}, I))$.
- *For any* $1 \leq q \leq r$*, there are* $\alpha \neq \beta \in \mathcal{M}_\lambda$ *such that if* $\mathsf{m}_{q-1}(i,k) \neq \mathsf{m}_q(i,k)$ *then* $\mathsf{m}_{q-1}(i,k) = \alpha$ *and* $\mathsf{m}_q(i,k) = \beta$.

Proof. We show how to construct $\mathsf{m}_0, \ldots, \mathsf{m}_r$. First we set $\mathsf{m}_0 = \mathsf{m}$. By Lemma A.3, it holds that $\mathsf{m}^*(i,k) = \mathsf{m}^*(j,\ell) \Rightarrow \mathsf{m}_0(i,k) = \mathsf{m}_0(j,\ell)$. If also the direction $\Leftarrow$ holds, then m_0 and m^* are equivalent and we are done. Otherwise there exist $i', j' \in [t]$, $k' \in [n_{i'}]$ and $\ell' \in [n_{j'}]$ such that $\mathsf{m}_0(i',k') = \mathsf{m}_0(j',\ell')$ but $\mathsf{m}^*(i',k') \neq \mathsf{m}^*(j',\ell')$. Take an arbitrary[4] $\beta \in \mathcal{M}_\lambda \setminus \mathsf{m}_0(V)$ (which exists by the assumption that $\mathcal{M} \geq \mathsf{size}(\mathsf{DB})$) and define m_1 by

$$\mathsf{m}_1(i,k) = \begin{cases} \beta & \text{if } \mathsf{m}^*(i,k) = \mathsf{m}^*(i',k') \\ \mathsf{m}_0(i,a) & \text{otherwise} \end{cases}$$

It is easy to see that $\mathsf{m}^*(i,k) = \mathsf{m}^*(j,\ell) \Rightarrow \mathsf{m}_1(i,k) = \mathsf{m}_1(j,\ell) \Rightarrow \mathsf{m}_0(i,k) = \mathsf{m}_0(j,\ell)$. From that along with the fact that m_0 and m^* are valid and conform with $\mathcal{L}(\mathsf{DB}, I)$, it is easily seen that m_1 is also valid and conforms with $\mathcal{L}(\mathsf{DB}, I)$. Now we repeat this process with respect to m_1 and m^* to get m_2, and so on. Since in the qth stage we have $|\mathsf{m}_q(V)| = |\mathsf{m}_{q-1}(V)| + 1$ and the image of m_q cannot be bigger than $|V| = \mathsf{size}(\mathsf{DB})$, the process must stop after $r \leq \mathsf{size}(\mathsf{DB})$ steps and we end up with m_r that is equivalent to m^*. ■

Lemma A.5. *Let* DB_0 *and* DB_1 *be two databases, and let* I *be a list of pairs of indices, such that* $\mathcal{L}(\mathsf{DB}_0, I) = \mathcal{L}(\mathsf{DB}_1, I)$*, with message maps* $\mathsf{m}, \mathsf{m}' : V \to \mathcal{M}_\lambda$*, respectively. Denote the common leakage by*

$$\mathcal{L} = (\mathsf{dim} = (t, (n_i)_{i=1}^t), \mathsf{Cols}, I = (i_a, j_a)_{a=1}^s, (J_a)_{a=1}^s)$$

and the common size by $\mathsf{size} = \sum_{i=1}^t n_i$. *Suppose that* $|\mathcal{M}_\lambda| \geq \mathsf{size} + 1$. *Then, we can efficiently construct a sequence of message maps* $\mathsf{m}_0, \ldots, \mathsf{m}_r$ *with the following properties:*

[4] As before, this can be done under the assumption that $\mathcal{M}_\lambda$ is efficiently enumerable.

- $r \leq 4 \cdot \mathsf{size}$.
- *For all* $0 \leq q \leq r$, $\mathsf{m}_q : V \to \mathcal{M}_\lambda$ *is valid and conforms with* $\mathcal{L}$.
- $\mathsf{m}_0 = \mathsf{m}$ *and* $\mathsf{m}_r = \mathsf{m}'$.
- *For any* $1 \leq q \leq r$*, there are* $\alpha \neq \beta \in \mathcal{M}_\lambda$ *such that if* $\mathsf{m}_{q-1}(i,k) \neq \mathsf{m}_q(i,k)$ *then* $\mathsf{m}_{q-1}(i,k) = \alpha$ *and* $\mathsf{m}_q(i,k) = \beta$.

Proof. We define an *$\mathcal{L}$-sequence between* m *and* m' *of length* r to be a sequence of messages maps $\mathsf{m}_0, \ldots, \mathsf{m}_r$ with the following properties:

- For all $0 \leq q \leq r$, $\mathsf{m}_q : V \to \mathcal{M}_\lambda$ is valid and conforms with $\mathcal{L}$.
- $\mathsf{m}_0 = \mathsf{m}$ and $\mathsf{m}_r = \mathsf{m}'$.
- For any $1 \leq q \leq r$, there are $\alpha \neq \beta \in \mathcal{M}_\lambda$ such that if $\mathsf{m}_{q-1}(i,k) \neq \mathsf{m}_q(i,k)$ then $\mathsf{m}_{q-1}(i,k) = \alpha$ and $\mathsf{m}_q(i,k) = \beta$.

Note that we can reverse such a sequence, and get an $\mathcal{L}$-sequence between m' and m. Also, we can concatenate an $\mathcal{L}$-sequence between m and m' of length r, and an $\mathcal{L}$-sequence between m' and m'' of length r', into an $\mathcal{L}$-sequence between m and m'' of length $r + r'$. By Claim A.4, we can construct an $\mathcal{L}$-sequence of length at most size between m and $\widehat{\mathsf{m}}$, where $\widehat{\mathsf{m}}$ is equivalent to m^*, and an $\mathcal{L}$-sequence of length at most size between m' and $\widehat{\mathsf{m}}'$, where $\widehat{\mathsf{m}}'$ is equivalent to m^*, thus is equivalent to $\widehat{\mathsf{m}}$. So it remain to show that we can construct an $\mathcal{L}$-sequence of length at most $2 \cdot \mathsf{size}$ between $\widehat{\mathsf{m}}$ and $\widehat{\mathsf{m}}'$, and the lemma follows from the above reverse and concatenation operations.

Let $\pi : \widehat{\mathsf{m}}(V) \to \mathcal{M}_\lambda$ be an injective map such that $\widehat{\mathsf{m}}' = \pi \circ \widehat{\mathsf{m}}$. Let $\alpha \in \widehat{\mathsf{m}}(V)$ and $\beta = \pi(\alpha)$ such that $\alpha \neq \beta$ (if no such α exists then $\widehat{\mathsf{m}} = \widehat{\mathsf{m}}'$ and we are done), and let $\gamma \in \mathcal{M}_\lambda \setminus \widehat{\mathsf{m}}(V)$ which must exist by the assumption that $|\mathcal{M}_\lambda| \geq \mathsf{size} + 1$. We define $\widehat{\mathsf{m}}_1$ and $\widehat{\mathsf{m}}_2$ as follows

$$\widehat{\mathsf{m}}_1(i,k) = \begin{cases} \gamma & \text{if } \widehat{\mathsf{m}}(i,k) = \beta \\ \widehat{\mathsf{m}}(i,k) & \text{otherwise} \end{cases}$$

$$\widehat{\mathsf{m}}_2(i,k) = \begin{cases} \beta & \text{if } \widehat{\mathsf{m}}(i,k) = \alpha \\ \gamma & \text{if } \widehat{\mathsf{m}}(i,k) = \beta \\ \widehat{\mathsf{m}}(i,k) & \text{otherwise} \end{cases},$$

Namely, in $\widehat{\mathsf{m}}_1$ we "free" the value β by replacing it with γ, and in $\widehat{\mathsf{m}}_2$ we replace α with β. $\widehat{\mathsf{m}}_2$ is equivalent to $\widehat{\mathsf{m}}'$ and agrees with it on the value β. Note that $\widehat{\mathsf{m}}$ do not agree with $\widehat{\mathsf{m}}'$ on the value β, since otherwise we would get $\pi(\alpha) = \pi(\beta)$ in contradiction to the injective property of π, so $\widehat{\mathsf{m}}_2$ also agrees with $\widehat{\mathsf{m}}'$ on any value that $\widehat{\mathsf{m}}$ agrees with $\widehat{\mathsf{m}}'$ on. Repeating this process with respect to $\widehat{\mathsf{m}}_2$ and $\widehat{\mathsf{m}}'$ yields $\widehat{\mathsf{m}}_3$ and $\widehat{\mathsf{m}}_4$ such that $\widehat{\mathsf{m}}_4$ agrees with $\widehat{\mathsf{m}}'$ on another value, etc. After at most size steps we get to $\widehat{\mathsf{m}}'$, and obtain a sequence of length at most $2 \cdot \mathsf{size}$ as wanted.

■

A.3 Proof of Claim 4.3

Let $\mathcal{A}$ be a valid adversary to the non-adaptive IND-security experiment. Denote the challenge phase algorithm of $\mathcal{A}$ by $\mathcal{A}^{(1)}(\mathsf{params}) \to (\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state})$, and the output phase algorithm by $\mathcal{A}^{(2)}(\mathsf{state}, \mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_b), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_b, I)) \to \sigma$. We define an adversary $\mathcal{B}$ to the $\mathsf{3Partition}$-security experiment as follows. In the setup phase, on input public parameters params, the adversary $\mathcal{B}$ works as follows:

- $\mathcal{B}$ computes $\mathcal{A}^{(1)}(\mathsf{params}) \to (\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state})$. Let m and m' be the message map of DB_0 and DB_1 respectively.
- $\mathcal{B}$ constructs message maps $\mathsf{m}_0, \ldots, \mathsf{m}_r$ as in Lemma A.5[5], such that $\mathsf{m}_0 = \mathsf{m}$ and $\mathsf{m}_r = \mathsf{m}'$.
- $\mathcal{B}$ samples $q \leftarrow \{1, \ldots, 4 \cdot p(\lambda)\}$, where $p(\lambda)$ is a bound on $\mathsf{size}(\mathsf{DB}_0) = \mathsf{size}(\mathsf{DB}_1)$.

Now, if $q > r$, then $\mathcal{B}$ skips to the output phase and simply outputs 0. Otherwise, let $\alpha, \beta \in \mathcal{M}$ such that if $\mathsf{m}_{q-1}(i,k) \neq \mathsf{m}_q(i,k)$ then $\mathsf{m}_{q-1}(i,k) = \alpha$ and $\mathsf{m}_q(i,k) = \beta$. Using $\mathsf{AddColumn}$ queries, $\mathcal{B}$ declares the following column sets:

$$\begin{aligned}
L &= \{\mathsf{col}_i \in \mathsf{Cols} | \exists k \quad \mathsf{m}_{q-1}(i,k) = \mathsf{m}_q(i,k) \in \{\alpha, \beta\}\} \\
R &= \{\mathsf{col}_i \in \mathsf{Cols} | \exists k \quad \mathsf{m}_{q-1}(i,k) \neq \mathsf{m}_q(i,k)\} \\
M &= \mathsf{Cols} \setminus (L \cup R)
\end{aligned}$$

It should be noted that L and R are disjoint, since otherwise it would imply that there are i and $k \neq \ell$ such that either $\mathsf{m}_{q-1}(i,k) = \alpha = \mathsf{m}_{q-1}(i,\ell)$ or $\mathsf{m}_q(i,k) = \beta = \mathsf{m}_q(i,\ell)$, in contradiction to the validity of m_{q-1} and m_q. Next, for each i and k such that $\mathsf{m}_{q-1}(i,k) = \mathsf{m}_q(i,k)$, using Enc queries, $\mathcal{B}$ retrieves the encoding of $\mathsf{m}_q(i,k)$ with respect to the column col_i. Also, for each $(i,j) \in I$, using $\mathsf{TokenGen}$ queries, $\mathcal{B}$ retrieves the tokens for the columns col_i and col_j. We argue that those queries are valid, namely, it never happens that $\mathsf{col}_i \in L$ and $\mathsf{col}_i \in R$, or vice versa. Otherwise, it would imply that there are $(i,j) \in I$, k and ℓ for which $(k,\ell) \in \mathsf{Join}_{\mathsf{DB}(m_{q-1})}(i,j) \triangle \mathsf{Join}_{\mathsf{DB}(m_q)}(i,j)$, where $\triangle$ denotes the symmetric difference of the sets, so $\mathcal{L}(\mathsf{DB}(\mathsf{m}_{q-1}), I) \neq \mathcal{L}(\mathsf{DB}(\mathsf{m}_q), I)$ in contradiction to the fact that both sides are equal to $\mathcal{L}(\mathsf{DB}_0, I)$. Lastly, $\mathcal{B}$ declare the challenge (α, β) and obtains the encryptions of α or β according to the columns in R. The encodings that $\mathcal{B}$ retrieves allow her to construct a database encoding E that is either $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_{q-1}))$ or $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_q))$, depending on whether $b = 0$ or $b = 1$, respectively, and the tokens that $\mathcal{B}$ retrieves allow her to construct $\boldsymbol{\tau} = \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_{q-1}), I) = \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_q), I)$. The final step of $\mathcal{B}$ is to compute $\mathcal{A}^{(2)}(\mathsf{state}, E, \boldsymbol{\tau}) \to \sigma$, and output σ. This completes the description of $\mathcal{B}$.

Fix some possible output $\mathsf{out} = (\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state})$ of $\mathcal{A}^{(1)}$, and denote by $\mathcal{E}_{\mathsf{out}}$ the event that $\mathcal{A}^{(1)}$ outputs out. This event can be seen as an event of both experiments $\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{A}}$ and $\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}$, since in both cases $\mathcal{A}^{(1)}$ is invoked exactly once. Note that fixing the output out in the experiment $\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}$ fixes the sequence $\mathsf{m}_0, \ldots, \mathsf{m}_r$. It holds that

$$\begin{aligned}
&\Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda, b) = 1 \middle| \mathcal{E}_{\mathsf{out}}\right] \\
&= \sum_{q=1}^{4\cdot p(\lambda)} \Pr[\mathcal{B} \text{ samples } q] \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda, b) = 1 \middle| \mathcal{E}_{\mathsf{out}}, \mathcal{B} \text{ samples } q\right] \\
&= \sum_{q=1}^{r} \Pr[\mathcal{B} \text{ samples } q] \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda, b) = 1 \middle| \mathcal{E}_{\mathsf{out}}, \mathcal{B} \text{ samples } q\right] \\
&= \sum_{q=1}^{r} \frac{1}{4 \cdot p(\lambda)} \Pr\left[\mathcal{A}^{(2)}(\mathsf{state}, E_{q-1+b}) = 1\right],
\end{aligned}$$

[5]Note that Lemma A.5 has the requirement that $|\mathcal{M}_\lambda| \geq \mathsf{size}(\mathsf{DB}_0) + 1$, however, by our assumption that $|\mathcal{M}_\lambda|$ is super-polynomial, for any polynomial-time adversary $\mathcal{A}$, this inequality holds for all sufficiently large λ.

where $E_{q-1+b} = (\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_{q-1+b})), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}(\mathsf{m}_{q-1+b}), I))$. Therefore,

$$\begin{aligned}
&\Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda,0)=1\middle|\mathcal{E}_{\mathsf{out}}\right] - \Pr\left[\mathsf{Exp}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda,1)=1\middle|\mathcal{E}_{\mathsf{out}}\right] \\
&= \frac{1}{4\cdot p(\lambda)}\cdot\Big(\Pr\left[\mathcal{A}^{(2)}(\mathsf{state}, \mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_0), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_0, I))=1\right] \\
&\qquad - \Pr\left[\mathcal{A}^{(2)}(\mathsf{state}, \mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_1), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_1, I))=1\right]\Big) \\
&= \frac{1}{4\cdot p(\lambda)}\cdot\Big(\Pr\left[\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{A}}(\lambda,0)=1\middle|\mathcal{E}_{\mathsf{out}}\right] - \Pr\left[\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{A}}(\lambda,1)=1\middle|\mathcal{E}_{\mathsf{out}}\right]\Big),
\end{aligned}$$

and deduce that

$$\mathsf{Adv}^{\mathsf{naIND}}_{\Pi,\mathcal{A}}(\lambda) = 4p(\lambda)\cdot\mathsf{Adv}^{\mathsf{3Par}}_{\Pi,\mathcal{B}}(\lambda). \tag{A.7}$$

For a probabilistic polynomial-time adversary $\mathcal{A}$, the bound $p(\lambda)$ on $\mathsf{size}(\mathsf{DB}_0)$ can be taken to be a polynomial, and the probabilistic adversary $\mathcal{B}$ runs in polynomial time. By the assumption that Π is $\mathsf{3Partition}$-secure, it holds that the RHS of Equation (A.7) is negligible, hence the LHS is negligible, and this shows that Π is non-adaptively IND-secure as claimed.

A.4 Proof of Claim 4.7

Proving that non-adaptive SIM security implies non-adaptive IND security is standard (here the non-adaptivity does not play a significant role): The adversary cannot distinguish between DB_0 and the simulation, and between the simulation and DB_1, hence cannot distinguish between DB_0 and DB_1.

For proving that non-adaptive IND security implies non-adaptive SIM security, our main observation is the one stated in Lemma A.3 (see Section A.2 for the proof), namely, that given the leakage $\mathcal{L}(\mathsf{DB}, I)$ we can efficiently produce a "canonical" database and use its encoding for the simulation.

Proof of Claim 4.7. We first prove that non-adaptive SIM security implies non-adaptive IND security. Let $\mathcal{B}$ be a valid probabilistic polynomial-time adversary to the non-adaptive IND-security experiment. We denote the algorithm of the setup and challenge phases by $\mathcal{B}^{(1)}(1^\lambda, \mathsf{params}) \to (\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state})$, where state is an internal state of $\mathcal{B}$, and the algorithm of the output phase by $\mathcal{B}^{(2)}(\mathsf{state}, E) \to \sigma$, where E is the input from the challenger (i.e., an encoding of DB_0 or DB_1, and the tokens).

We build an adversary $\mathcal{A}$ to the non-adaptive SIM-security experiment as follows: Again we denote the two algorithms $\mathcal{A}$ by $\mathcal{A}^{(1)}(1^\lambda, \mathsf{params}) \to (\mathsf{DB}, I, \mathsf{state})$ and $\mathcal{A}^{(2)}(\mathsf{state}, E) \to \sigma$. $\mathcal{A}^{(1)}$ runs $(\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state}) \leftarrow \mathcal{B}^{(1)}(1^\lambda)$, samples $c \leftarrow \{0,1\}$ and outputs $(\mathsf{DB}_c, I, \mathsf{state})$. $\mathcal{A}^{(2)}$ runs $\mathcal{B}^{(2)}(\mathsf{state}, E) \to \sigma$ and output $\sigma \oplus c$. For any simulator $\mathcal{S}$ we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda) &= \left|\Pr[\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda,0)=1] - \Pr[\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda,1)=1]\right| \\
&= \left|\Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=1|c=0] - \Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=0|c=1]\right| \\
&= \Big|\Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=1|c=0] - \Pr[\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)=1|c=0] \\
&\quad + \Pr[\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)=0|c=1] - \Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=0|c=1]\Big| \\
&= \Big|\Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=1|c=0] - \Pr[\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)=1|c=0] \\
&\quad + \Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)=1|c=1] - \Pr[\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)=1|c=1]\Big| \\
&= 2\cdot\mathsf{Adv}^{\mathsf{naSIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda),
\end{aligned}$$

where we used the fact

$$\Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda) = 1 | c = 0] = \Pr[\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda) = 0 | c = 1],$$

which follows from the fact that in both conditional spaces $\mathcal{A}^{(1)}$ output the same distribution $(\mathcal{L}(\mathsf{DB}, I), \mathsf{state})$, and that $\mathcal{A}^{(2)}$ inverts the output of $\mathcal{B}^{(2)}$ in case that $c = 1$. By the assumption that Π is non-adaptively SIM-secure there exists a simulator $\mathcal{S}$ for which $\mathsf{Adv}^{\mathsf{naSIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ is negligible, therefore $\mathsf{Adv}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda)$ is negligible as claimed.

We now prove that non-adaptive IND security implies non-adaptive SIM security. We define a simulator $\mathcal{S} = (\mathcal{S}^{(1)}, \mathcal{S}^{(2)})$ as follows: $\mathcal{S}^{(1)}$ runs $(\mathsf{sk}, \mathsf{params}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and outputs $(\mathsf{state}, \mathsf{params}) = ((\mathsf{sk}, \mathsf{params}), \mathsf{params})$. Given $\mathcal{L}$ and $\mathsf{state} = (\mathsf{sk}, \mathsf{params})$, $\mathcal{S}^{(2)}$ computes[6] $\mathsf{DB}^* = \mathsf{ConstructDB}(\mathcal{L})$, and outputs $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}^*)$ and $\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}^*, I)$.

Let $\mathcal{A}$ be a probabilistic polynomial-time adversary to the non-adaptive SIM-security experiment. We construct a valid adversary $\mathcal{B}$ to the non-adaptive IND-security experiment as follows: $\mathcal{B}^{(1)}$ runs $(\mathsf{DB}, I, \mathsf{state}) \leftarrow \mathcal{A}^{(1)}(1^\lambda, \mathsf{params})$, calculates $\mathsf{DB}^* = \mathsf{ConstructDB}(\mathcal{L}(\mathsf{DB}, I))$, and outputs $(\mathsf{DB}_0, \mathsf{DB}_1, I, \mathsf{state}) = (\mathsf{DB}, \mathsf{DB}^*, I, \mathsf{state})$. $\mathcal{B}^{(2)}$ gets as input state, $\mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_b)$ and $\mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_b, I)$, runs $\sigma \leftarrow \mathcal{A}^{(2)}(\mathsf{state}, \mathsf{Enc}_{\mathsf{sk}}(\mathsf{DB}_b), \mathsf{TokenGen}_{\mathsf{sk}}(\mathsf{DB}_b, I))$, and outputs σ. The experiments $\mathsf{Real}^{\mathsf{naSIM}}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda, 0)$ behave in the same way, and also the experiments $\mathsf{Ideal}^{\mathsf{naSIM}}_{\mathcal{A},\mathcal{S}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda, 1)$ behave in the same way. So it follows that $\mathsf{Adv}^{\mathsf{naSIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda) = \mathsf{Adv}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda)$. By the assumption that Π is non-adaptively IND-secure it follows that $\mathsf{Adv}^{\mathsf{naIND}}_{\Pi,\mathcal{B}}(\lambda)$ is negligible, hence $\mathsf{Adv}^{\mathsf{naSIM}}_{\Pi,\mathcal{A},\mathcal{S}}(\lambda)$ is negligible as claimed. ■

[6]Note that Lemma A.3 has the requirement that $|\mathcal{M}_\lambda| \geq \mathsf{size}(\mathsf{DB})$, however, by our assumption that $|\mathcal{M}_\lambda|$ is super-polynomial, for any polynomial-time adversary $\mathcal{A}$, this inequality holds for all sufficiently large λ.