Structural Composition Attacks on Anonymized Data

Tobias Nilges, Jörn Müller-Quade
Forschungsgruppe Kryptographie und Sicherheit
Institut für Theoretische Informatik
Karlsruher Institut für Technologie
Am Fasanengarten 5
76131 Karlsruhe
surname@kit.edu

Matthias Huber
Sichere Software und Kryptographie
FZI Forschungszentrum Informatik
Haid-und-Neu-Strae 10-14
76131 Karlsruhe
mhuber@fzi.de

Abstract: Anonymized releases of databases are increasingly available in public. The composition of two releases does not necessarily fulfil any anonymity notion, even if the individual releases do fulfil that anonymity notion on their own.

In this paper, we study composition scenarios and provide formalizations. We introduce a formal framework to study the composition of databases on a structural level and show the equivalence of the composition scenarios used in the literature. We show that known attacks on anonymity notions can be reduced to two simple properties and only need limited side information.
keywords: database anonymization, data composition, anonymity notions

1 Introduction

Anonymity has been in focus of research on privacy-preserving database disclosure [Swe02, Dwo06, FWCY10] over the past years. The goal is to publish an anonymized database in order to allow others to analyze it while preserving the privacy of the individuals represented in the database. For this scenario *anonymity notions* were introduced. Their purpose is to formally express the limitation of information disclosure from the database or guarantees provided by the anonymization. The property k-anonymity [SS98], for example, states that for any individual there have to be at least k (or zero) rows in the released database that can be associated with the individual.

A huge challenge is that the composition of two releases, i. e. the combination of two (independently) anonymized datasets, does not necessarily fulfil any anonymity notion [GKS08, Swe02]. Nevertheless, multiple releases from different sources as well as different releases from the same source are common in practice. This highlights the need to formally define

database composition in such a way that this problem can be considered in the design of anonymity notions. No formal investigation of database composition, however, has been carried out. Additionally, there are no results based on the structure instead of concrete attribute values.

Another challenge in privacy-preserving database disclosure is modeling background knowledge. Background knowledge (or side information) is knowledge an adversary obtains independently from other data sources. Background knowledge can be modeled as a database release. Therefore, many side-information attacks on the anonymity of database releases can be considered composition attacks. In our work, we show an interesting connection between modeling background knowledge and the composition of anonymized database releases. Because modeling background knowledge has proven to be difficult, it was only conducted for a small class of anonymization procedures [MKM+07, CRL07, WFWP07]. Our framework gives way to new approaches for considering background knowledge.

Our Contribution. In this paper, we provide the first formal definition of database composition by extracting two scenarios from attacks presented in literature [Swe02, GKS08, NS08, MGKV06, LLV07]. We prove that these scenarios cover all possible forms of database composition and that they can be transformed into each other, thereby allowing for considering just one scenario during the design of anonymity notions.

We then move on to show that, to the best of our knowledge, most composition attacks (in fact all of the above mentioned attacks) can be reduced to two properties of anonymity procedures, namely *Locatability* and *Exact Sensitive Value Disclosure* (ESVD), first proposed by Ganta et al. [GKS08]. Locatability allows an attacker to determine a set of possible pre-images of an anonymized value including the correct value, while ESVD means that the anonymization procedure does not alter the sensitive values, i.e. keeps them as is in the anonymized database.

In order to control background knowledge in composition scenarios, we introduce the notion of symbolic databases. Although these databases are devoid of semantic information leaving merely the structure of the database, Locatability and ESVD still exist for symbolic databases. For some notions, no side information is necessary to carry out a composition attack. We show that ESVD suffices to break k-anonymity on a structural level. This provides evidence that notions implying Locatability and ESVD cannot provide an unconditional security guarantee if several anonymized databases are combined.

Related Work. In the context of database anonymization two lines of research have been established. On the one hand partition based anonymity notions such as k-anonymity [SS98], l-diversity [MGKV06] and t-closeness [LLV07], have been thoroughly analyzed, including analysis of the complexity of the methods [MW04, JA05, LDR05, ZYW05] and proposed attacks [XT06, LLV07, GKS08].

The other line of research examines adding noise to databases [PS86, Kim86, DP91, WdW96]. The main result is differential privacy [Dwo06]. There are several relaxations of this notion [MT07, DKM+06, NRS07, MPRV09] as well as some variations [BBG+11, MPRV09, KM12]. Recently, Gehrke et al. [GLP11] presented an attack against differential privacy.

Composition of anonymized databases has been investigated mainly in the context of differential privacy [DMNS06, DKM+06, GKS08, KM12], while partition based anonymity schemes are only explicitly considered by Ganta et al. [GKS08]. Nevertheless, attacks as presented by [MGKV06, LLV07, NS08] are also composition attacks.

Since differential privacy is secure under arbitrary background knowledge [Dwo06, GKS08], modeling background knowledge has been mainly examined in the area of partition-based anonymity schemes [MKM+07, CRL07, WFWP07].

Outline of the paper. This paper is structured as follows: In the next section, we define the basic concepts used throughout this paper. In Section 3, we introduce the concept of *symbolic databases* and present a framework around this notion. In Section 4, we present our main results: the equivalence of different composition scenarios and the interpretation of known attacks as specific exploits of two properties on a structural level. Section 5 concludes.

2 Preliminaries

In this section, we present definitions of the concepts used throughout this paper. After a formal definition of databases and quasi-identifiers we present three database composition scenarios. Then, we define generalizations of Locatability and Exact Sensitive Value Disclosure (cf. Ganta et al. [GKS08]).

2.1 Notation

Let a *database* $d = \{t_1, t_2, \ldots, t_n\}$ be a set of tuples with attributes $A = \{A_1, A_2, \ldots, A_m\}$. We denote the value of attribute A_i in tuple t_j by $t_j[A_i]$. Throughout this paper, we use the terms tuple and row interchangeably. We call the set of all i-th elements of each tuple in d the i-th column of d and $|d| = n * m$ the size of d.

Throughout this work, we use the operators projection π and selection σ as follows: Let $b = \{b_1, \ldots, b_l\} \subseteq A, l \leq m$, be a set of attribute values of d. Then, π projects a database d to a database d' that only contains a subset of the attributes of d:

$$\pi_b(d) = \cup_{j=1..n}(t_j[b_1], \ldots, t_j[b_l])$$

Likewise, σ selects a number of rows of d fulfilling some condition of the form:

$$\sigma_{a=v} = \{t \in d \mid t[a] = v\},$$

where $a \in A$ is an attribute and v is a value of the domain of A.

A set of non-sensitive attributes $Q = \{Q_1, \ldots, Q_w\} \subseteq A$ is called a *quasi-identifier* if these attributes serve to uniquely identify at least one individual in the database. Let $\mathsf{S} \subset A \setminus Q$ be the set of sensitive attributes, w.l.o.g. we only consider the case of $|\mathsf{S}| = 1$.

We abbreviate a probabilistic polynomial-time turing machine by PPT without explicitly describing the random input. In order to capture deterministic anonymization functions as well as probabilistic anonymization mechanisms, througout this paper, we talk about *anonymization mechanisms*.

2.2 Anonymity Notions

In order to describe the level of anonymity provided by an anonymization, anonymity notions have been defined. Many notions such as k-anonymity and notions based thereon describe the result of the anonymization mechanism. Notions like differential privacy describe the anonymization mechanism itself with respect to bounded [MPRV09] and unbounded [Dwo06] adversaries.

We stress that despite all anonymity notions seem to have a standard method in order to achieve them (e. g. adding noise for differential privacy, or to coarsen attribute values for k-anonymity), we have to distinguish between methods and guarantees. For example, another approach to achieve differential privacy is presented in [BBG$^+$11], without the addition of noise, but by assuming that the database already has some additional entropy for the adversary. On the other hand, the addition of Laplace noise does not yield differentially private methods in general, but only under certain assumptions about the distribution of attribute values.

In the following we assume an anonymity notion to state a form of guarantee $\mathcal{P}$, which can be achieved by a (probabilistic) anonymization mechanism f. We say f realizes $\mathcal{P}_k$, or $f \in \mathcal{P}_k$, where k is a privacy parameter or a set of privacy parameters.

2.3 Differential Privacy

We will briefly review the definition of differential privacy, since we will later use it to motivate our composition notions.

Definition 1 ([Dwo06]) *A randomized function f gives ε-differential privacy if for all data sets d_1 and d_2 differing on at most one element, and all $S \subseteq Range(f)$,*

$$\Pr[f(d_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[f(d_2) \in S]$$

The intuition behind this is that an adversary cannot discern two databases (except for a small factor) that differs in a single entry, i.e. by the result of the mechanism it is nearly impossible to say if a certain entry is in the database or not. Differential privacy can be achieved by adding noise to the output, thereby hiding the true value. The noise distribution has to be Laplacian with variance $\frac{1}{\varepsilon}$.

2.4 Locatability and Exact Sensitive Value Disclosure

Locatability and Exact Sensitive Value Disclosure were first presented by Ganta et al. [GKS08]. While they use these notions only in the context of their intersection attack (cf. Section 2.5), we will present a generalized definition of their notions. Later, we will show how these properties translate to properties for symbolic databases and lead to an interesting type of attacks.

Definition 2 *Let f be an anonymization mechanism. Then we say f has* Locatability *iff there exists an adversary $\mathcal{A}$, a database d, a quasi-identifier Q and a tuple $t \in d$, such that $\mathcal{A}(\pi_Q(t))$ returns a strict subset p of $d' = f(d)$ with $f(t) \in p$.*

This definition of Locatability is weak in the sense that it can be very easy to achieve. Locatability only improves an attack if the distribution of sensitive values in the resulting partition differs from the distribution of sensitive values in the database. Consider for instance a database that has t-closeness [LLV07] with $t = 0$ for every sensitive attribute, i.e. the distribution of every partition is the same as in the complete database. Then Locatability is of no use to an adversary, however, achieving $t = 0$ in real databases is virtually impossible. In contrast, in the case of k-anonymity, an adversary can definitely identify the bucket of any individual [Swe02, GKS08].

The advantage of our generalization of Locatability from k-anonymity to general anonymity notions is that it enables us to highlight the similarity of many known composition attacks [GKS08, GLP11, NS08]. Additionally, we provide a precise formulation for identifying a set in the anonymized database that corresponds to an individual.

We now define Exact Sensitive Value Disclosure (ESVD).

Definition 3 *Let f be an anonymization mechanism. Then we say f has Exact Sensitive Value Disclosure iff at least one sensitive value of d is contained in $f(d)$. That is,*

$$\exists S \in \mathsf{S}, \exists v : v \in \pi_S(d) \wedge v \in \pi_S(f(d))$$

Recall that S describes the set of sensitive values. By removing the assumption that the anonymization mechanism is partition-based and possibly contains Locatability, we can prove that ESVD generally poses a threat during composition, even without background knowledge (cf. Section 4.4).

2.5 Intersection Attack

Algorithm 1 describes the intersection attack of Ganta et al. [GKS08]. As already mentioned, this attack heavily relies on ESVD and Locatability. First, the Locatability property is used to obtain a set of database entries associated with an individual of the overlapping population of the releases. Then, the sensitive values are extracted for each anonymized release. By intersecting these sets, one gets either the sensitive value associated with an

individual or a relatively small set of values. According to the measurements of Ganta et al., they can achieve a success rate of up to 60% based on real census data.

Algorithm 1 intersection attack (from [GKS08]).

1: $R_1, \ldots, R_n \leftarrow n$ independent anonymized releases
2: $P \leftarrow$ set of overlapping population
3: **for** each individual i in P **do**
4: **for** $j = 1$ to n **do**
5: $e_{ij} \leftarrow$ Get_equivalence_class(R_j, i)
6: $s_{ij} \leftarrow$ Sensitive_value_set(e_{ij})
7: **end for**
8: $S_i \leftarrow s_{i1} \cap s_{i2} \cap \ldots \cap s_{in}$
9: **end for**
10: **return** $S_1, \ldots, S_{|P|}$

In the algorithm, the function *Get_equivalence_class()* returns the partition of the database which contains the individual. By applying the function *Sensitive_value_set()* to the result of *Get_equivalence_class()* the sensitive values are extracted from the partitions.

In Section 4.3 we will show how this attack can be carried out with symbolic databases.

3 The Symbolic Framework

It has proven to be difficult to model or control arbitrary side information of an adversary in the context of database privacy [MKM+07, CRL07, WFWP07]. Our approach to this problem first eliminates all side information and later adds the information that cannot be hidden in the real world.

Our model is meaningful in the following sense: Successfully performing an attack in our model yields a successful attack in the real world. Thus, our ability to model ESVD and Locatability and to reduce attacks to these two properties in our framework also implies that the problems that arise from these properies are retained in the real world.

In this section, we first define the building blocks of our framework. To illustrate the relevance of this framework, in the following sections we show that (a) the composition notions presented in Section 4.1 can be proven as equivalent, (b) many composition attacks can be simulated in the symbolic framework (cf. [GKS08, GLP11, Swe02, NS08]) and, therefore, (c) do only need limited side information quantifiable in our model.

3.1 Representation of Databases

As a first step we define the class of structurally identical databases. In order to study database composition on a structural level, we replace all attribute values with meaning-

less symbols. This renders side information useless. Informally, structurally identical databases are a set of databases that can all be projected onto the same symbolic database.

Definition 4 $\mathfrak{S}(d) = \{d' | \exists w_r : w_r(d') = d\}$ *with* w_r *being a wrapper function that applies a bijective function* w_i *to each column* i *of* d*. We call* $\mathfrak{S}(d)$ *the databases* structurally identical *to* d*.*

$\mathfrak{S}$ implies an equivalence relation on databases. Instead of $d' \in \mathfrak{S}(d)$ we write $d \overset{\mathfrak{S}}{=} d'$ and call them structurally identical. Note that this notion also includes all permutations of a database, which is due to the definition of a database as a set.

Now, we can replace the values of any representative of $\mathfrak{S}(d)$ to create a symbolic database. The notion of symbolic databases is defined w. r. t. an adversary. In the following experiment, the adversary tries to relate a given symbolization of a database to one of two structurally identical databases she chose. The game is related to known security definitions for encryption schemes such as IND-CPA or IND-CCA2. In this experiment, we write $\leftarrow$ insted of $=$ in order to indicate that the allocation can involve a probabilistic mechanism:

Definition 5 *Let* $\mathcal{A}$ *be an adversary,* $d \in DB$*,* $i \in \{0, 1\}$*,* $\Sigma : DB \to DB$ *an injective function. The experiment* $Symb_{\mathcal{A}}^{i,\Sigma}(d)$ *is defined as follows:*

$d_0, d_1 \leftarrow \mathcal{A}(\mathfrak{S}(d))$
$d_0 \overset{\mathfrak{S}}{\neq} d_1$ *: return* $\perp$
$b \leftarrow \mathcal{A}(\Sigma(d_i))$
return $b \overset{?}{=} i$

Here Σ is a function that replaces the values of one of the two adversarially chosen databases with random symbols. This function is not known by the adversary. Then the adversary has to guess the pre-image of the symbolized database. With this experiment, we can define symbolic databases as follows:

Definition 6 *Let* Σ*:* $DB \to DB$ *be a bijective function. We call* Σ *a database symbolization function iff for all adversaries* $\mathcal{A}$ *and for all* $d \in DB$ *the following holds:*

$$\left| Pr[Symb_{\mathcal{A}}^{0,\Sigma}(d) = 1] - Pr[Symb_{\mathcal{A}}^{1,\Sigma}(d) = 1] \right| \leq \epsilon$$

for ϵ *negligible in* $|d|$*. We call* $\Sigma(d)$ *a* symbolic database *of* d*.*

Note that we use the definition above in order to define functions that change the attribute values of a given database rather than its structure.

In order to allow for composition scenarios, we use the same function that replaces the attribute values for corresponding columns in different databases. The definition of symbolic databases implies that attribute names get symbolized as well.

As an example for the symbolization, consider the databases depicted in Figure 1.

130**	< 30	*	AIDS
130**	< 30	*	Heart Disease
130**	< 30	*	Viral Infection
130**	< 30	*	Viral Infection
130**	≥ 30	*	Cancer
130**	≥ 30	*	Heart Disease
130**	≥ 30	*	Viral Infection
130**	≥ 30	*	Viral Infection

(a)

a	b	e	f
a	b	e	g
a	b	e	h
a	b	e	h
a	c	e	i
a	c	e	g
a	c	e	h
a	c	e	h

(b)

Figure 1: An example database (a) with attribute names omitted and a representative of its symbolic databases (b): attribute values are replaced with symbols. The two databases have identical structure, but the symbolic database is stripped of meaning. Therefore, no background knowledge can be used to extract information from it.

In this simplified example, values are mapped to letters of the alphabet. The replacement is done in no particular order because, according to our definition, the database in Figure 1(a) is actually a set. Another injective function that can serve as a symbolization function is a hash function.

Remark. It is still possible to code information into a symbolic database, e. g. how often a certain symbol occurs in the original database. We will use this property later in order to show the equivalence of the composition notions.

Remark. By requiring a polynomially bounded adversary in the above definitions, symbolization functions such as hash functions or encryption functions can be used.

3.2 Anonymization Scenarios of Symbolic Databases

There are different possibilities for applying symbolization during anonymization. Consider Figure 2, where a database d gets anonymized to d' with the anonymization mechanism f. The database symbolization Σ can be applied prior to the anonymization, after it, or both.

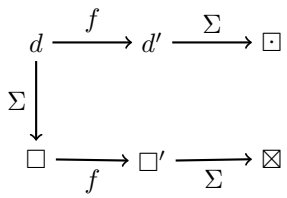

Figure 2: Anonymization scenarios for symbolic databases. d denotes the original database, f a database anonymization mechanism, Σ the database symbolization, and the different squares denote symbolic databases.

In general, even if the input of an anonymization mechanism is a symbolic database, this does not necessarily hold for the output. Consider for example an anonymization function that counts occurrences of attribute values. The output database then contains natural numbers that can be combined with background knowledge, e. g. that $3 > 2$. Thus we have to apply symbolization after anonymization.

We also want to apply the symbolization prior to anonymization for the following reasons: For one, this is the most general case for the examination of composition scenarios. If an anonymization does not compose in the most general case it also does not compose with additional knowledge. Another reason is that we do not want adversaries to exploit the Locatability oracle (i.e. use the oracle to win the game in Definition 5) which we define in the next section.

Therefore, we choose the following scenario: $d \xrightarrow{\Sigma} \square \xrightarrow{f} \square' \xrightarrow{\Sigma} \boxtimes$. We apply the symbolization Σ prior to anonymization and after it.

3.3 Locatability and ESVD for Symbolic Databases

We introduced Locatability and ESVD in Section 2.4. Since these properties can be exploited for composition attacks, in order to study composition attacks on a structural level, we need to be able to apply these properties to symbolic databases. In this section, we show how they translate into our framework.

Locatability. An adversary can always execute the anonymization mechanism f on a number of quasi-identifiers and compare the results to the anonymized database (although with new random coins, if the mechanism is probabilistic). This might enable an adversary to locate subsets in the anonymized database that relate to specific quasi-identifiers.

If we want an adversary to be able to do this with symbolic databases, we have to give her oracle access to $\Sigma \circ f$, where Σ is the symbolization and f the anonymization mechanism, since we do not want the adversary to know Σ. If f is probabilistic, the oracle has to use new random coins. Note that without the oracle, there is no Locatability for symbolic databases. Otherwise, the adversary would be able to invert Σ and thus win the game specified in Definition 5.

The notion of Locatability introduces a certain degree of side information into the symbolic framework, namely exactly the information that is necessary to obtain Locatability. However, the adversary still can not utilize her own background knowledge.

The construction of a Locatability oracle for an anonymity notion in the symbolic framework yields the background information necessary to perform Locatability for this notion in general, i. e. in the standard model. Thus the designer of an anonymity notion can check if the Locatability information is available in her setting.

ESVD. If an anonymization mechanism has the ESVD property, this property is kept in the symbolic framework, since we use one distinct symbolization for each column, and require that the same values in two corresponding columns are mapped to the same symbolic value.

With these definitions one can show that (a) the deanonymization of the Netflix dataset [NS08]

as well as (b) the attack of Gherke et al. [GLP11] is possible due to Locatability, (c) general attacks on k-anonymity presented in [Swe02] and in [GKS08] are possible due to ESVD and Locatability. These attacks can be simulated with symbolic databases where no other background knowledge than knowledge introduced by Locatability is available.

Locatability in the Netflix dataset stems from the sparsity of the columns representing movies, such that a set of movies identifies a set of individuals (cf. [NS08]). In the case of differentially private data [Dwo06], correlation between the entries can lead to Locatability. Gehrke et al. [GLP11] can locate an individual due to known associations with other individuals in the dataset and then derive the sensitive value. Although they do not use the term Locatability, their example is perfectly captured by our generalized definition.

In the next section, we will show a detailed example for an intersection attack with Locatability as well as one without Locatability.

4 Results in the Symbolic Framework

In this section we present our main results. We prove the equivalence of the composition scenarios (cf. Section 4.1) for symbolic databases, which translate straightforwardly to the plain model. Therefore, a proof of (non-)composability of an anonymity notion for a single composability scenario suffices for a general proof. Interestingly, due to the constructive nature of our proof, this also provides an adversary with a new attack. If her attack works out in one composition scenario, which was not considered for the anonymity notion, this attack can be converted by our technique to the other composition scenario.

We also show that known attacks such as the intersection attack of Ganta et al. (cf. Section 2.5) can be conducted with symbolic databases. We use a generalization of this attack and provide evidence that anonymization notions with ESVD cannot compose in general, even on a structural level with limited side information.

4.1 Composition Scenarios for Anonymized Databases

By analyzing the attacks presented in literature, e. g. [Swe02, GKS08, Dwo06, BBG+11], we derive two composition schemes, both of which can be generalized to a third scheme. In the first composition scheme an adversary is given two excerpts of different databases, which are anonymized by the same anonymization mechanism (cf. Figure 3(a)). The second scheme represents the case where an adversary has access to two different anonymizations of the same database, which are anonymized with different anonymization mechanisms (cf. Figure 3(b)). The third approach is the direct generalization of both composition scenarios, where two (possibly) different tables are anonymized by two (possibly) different anonymization mechanisms. This is depicted in Figure 3(c). Obviously, this captures every possible form of composition.

We motivate our definitions for secure composition as follows. An adversary $\mathcal{A}$ tries to

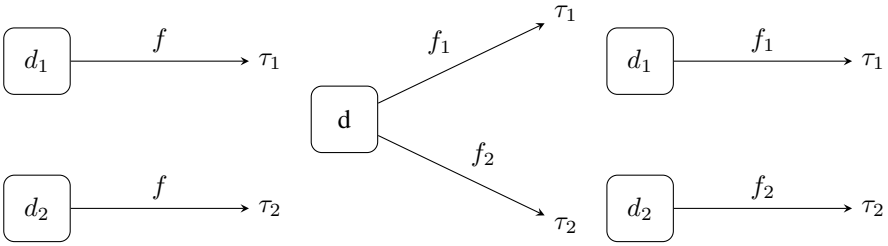

Figure 3: The database composition notion *Secure Composition under Independent Anonymizations and Databases* (SC-IAD) and its two specializations *Secure Composition under Independent Databases* (SC-ID) and *Secure Composition under Independent Anonymizations* (SC-IA). In Section 4 we show that all three definitions are equivalent.

learn a predicate p of the database d. For two anonymizations $f_1(d_1)$ and $f_2(d_2)$ all the adversary can learn without combining both anonymizations is bounded by

$$\max_{i\in\{0,1\}}\{\Pr[\mathcal{A}(f_i(d_i)) = p(d_i)]\}$$

Thus we have to define the security of a composition of anonymized databases with respect to what an adversary can learn without composition. It is common knowledge that the release of anonymized data yields some information to an adversary, because perfect anonymization leads to no utility (cf. e. g. [Dwo06]). To capture this problem we introduce a *degradation function* κ which bounds the degradation of anonymity relative to the privacy parameter.

Example 1 *We illustrate our approach by applying it to differential privacy. The composition of differential privacy was shown among others in [DKM+06] and falls into the scenario SC-IAD. Here f_1 and f_2 represent anonymization mechanisms for $\mathcal{P}_{\varepsilon_1}$ and $\mathcal{P}_{\varepsilon_2}$, respectively. This example is highly simplified, but it illustrates the intuition behind our definitions. The following has to hold for secure composition (cf. Definition 9):*

$$\Pr[\mathcal{A}(f_1(d_1), f_2(d_2)) = p(d_1 \cup d_2)] \leq \kappa(\varepsilon_1, \varepsilon_2) \cdot \max_{i\in\{0,1\}}\{\Pr[\mathcal{A}(f_i(d_i)) = p(d_i)]\} \quad (1)$$

The composition of differential privacy yields the privacy parameter $\varepsilon_1 + \varepsilon_2$ for the left hand side of 1, while we can bound the other side by a privacy parameter $\max\{\varepsilon_1, \varepsilon_2\}$. The highest probability of an adversaries' guess can be substituted on both sides of the equation.

$$e^{\varepsilon_1+\varepsilon_2} \cdot 2 \max_{i\in\{0,1\}}\{\Pr[f_i(d_i) \in S]\} \leq \kappa(\varepsilon_1, \varepsilon_2) \cdot e^{\max\{\varepsilon_1,\varepsilon_2\}} \cdot \max_{i\in\{0,1\}}\{\Pr[f_i(d_i) \in S]\}$$

$$\Leftrightarrow \qquad 2e^{\varepsilon_1+\varepsilon_2} \leq \kappa(\varepsilon_1, \varepsilon_2) \cdot e^{\max\{\varepsilon_1,\varepsilon_2\}}$$

This leads to a degradation function $\kappa(\varepsilon_1, \varepsilon_2) = 2e^{\max\{\varepsilon_1,\varepsilon_2\}}$ and shows that the composition of differential privacy can be intuitively expressed by our definition.

We now state the definitions for the secure composition of the above mentioned scenarios.

Definition 7 *An anonymity notion $\mathcal{P}$ composes* securely under independent databases *(SC-ID) iff for all anonymization mechanisms $f \in \mathcal{P}_k$ there exists a degradation function κ such that for all adversaries $\mathcal{A}$, for all databases d_1 and d_2 and for all predicates $p : \{0,1\}^* \to \{0,1\}$ the following holds:*

$$\Pr[\mathcal{A}(f(d_1), f(d_2)) = p(d_1 \cup d_2)] \leq \kappa(k) \cdot \max_{i \in \{0,1\}} \{\Pr[\mathcal{A}(f(d_i)) = p(d_i)]\}$$

Of course, composition is only a privacy threat if the databases d_1 and d_2 have some sort of correlation, e. g. an overlap of individuals. By independent databases we mean that the two databases do not explicitly cover the same population. If a database anonymity notion $\mathcal{P}$ is a property of the anonymization mechanism (e. g. differential privacy), we have to extend the definition of $\mathcal{P}$ and quantify over pairs of anonymized databases instead of single databases.

Definition 8 *An anonymity notion $\mathcal{P}$ composes* securely under independent anonymizations *(SC-IA) iff for all anonymization mechanisms $f_1 \in \mathcal{P}_{k_1}$ and $f_2 \in \mathcal{P}_{k_2}$ there exists a degradation function κ such that for all adversaries $\mathcal{A}$, for all databases d and for all predicates $p : \{0,1\}^* \to \{0,1\}$ the following holds:*

$$\Pr[\mathcal{A}(f_1(d), f_2(d)) = p(d)] \leq \kappa(k_1, k_2) \cdot \max_{i \in \{0,1\}} \{\Pr[\mathcal{A}(f_i(d)) = p(d)]\}$$

Definition 9 *An anonymity notion $\mathcal{P}$ composes* securely under independent anonymizations and databases *(SC-IAD) iff for all anonymization mechanisms $f_1 \in \mathcal{P}_{k_1}$ and $f_2 \in \mathcal{P}_{k_2}$ there exists a degradation function κ such that for all adversaries $\mathcal{A}$, for all databases d_1 and d_2 and for all predicates $p : \{0,1\}^* \to \{0,1\}$ the following holds:*

$$\Pr[\mathcal{A}(f_1(d_1), f_2(d_2)) = p(d_1 \cup d_2)] \leq \kappa(k_1, k_2) \cdot \max_{i \in \{0,1\}} \{\Pr[\mathcal{A}(f_i(d_i)) = p(d_i)]\}$$

Obviously, if an anonymity notion composes securely under independent anonymizations and databases (SC-IAD), it also composes securely under independent databases (SC-ID) and under independent anonymizations (SC-IA). In Section 4.2, we will show that this also holds the other way round.

4.2 Equivalence of Composition Notions

We provide a formal proof that the composition scenarios defined in Section 4.1 can be transformed into each other. While this result seems to be rather intuitive if one considers normal databases, it appears to be difficult to prove the equivalence. Thus we show that these scenarios can be transformed if one considers symbolic databases, which implies a general transformation for standard databases.

Theorem 1 *For symbolic databases, the composition scenarios SC-IAD, SC-ID, and SC-IA are equivalent in terms of polynomial-time transformation, i. e. each instance of a composition scenario can efficiently be simulated with an instance of a composition scenario of another class.*

Proof 1 *In the following we use SC-ID, SC-IA, and SC-IAD as sets that contain all corresponding instances of database composition. Since the transformations from SC-ID and SC-IA to SC-IAD are trivial, we only need to show transformations from SC-IAD to SC-ID and SC-IA, respectively. Because a database can be extended, e. g. by adding blank columns and rows, in the following proof, w.l.o.g. we only consider databases of the same size.*

In our transformations, we encode additional information in the databases. For each bit of information we want to encode, we add a new column to the database. Since for symbolic databases, the actual attribute values have no meaning (otherwise an adversary would be able to invert the symbolization) and databases have no specific order (we defined them as sets), we encode information into the new columns by filling them with identical or different symbols.

SC-IAD $\subseteq$ ***SC-ID:*** *Let* d_1, d_2 *be databases and* f_1, f_2 *database anonymization mechanism of an SC-IAD instance. SC-ID instances require its two databases to be anonymized by the same anonymization mechanism* f*. We need to specify an algorithm* $g : DB \times DB \rightarrow DB \times DB$ *and an anonymization mechanism* f *with* $f(g(d_1, d_2)[1]) = f_1(d_1)$ *and* $f(g(d_1, d_2)[2]) f_2(d_2)$ *where* $[x]$ *selects the* x*-th element of the preceding tuple.*

We design g *such that it adds one additional column to both of its input databases. The column is filled according to the following scheme: For* d_1*,* g *fills the additional column of each tuple in* d_1 *with identical values. This encodes a* 1*. In the additional column of* d_2*,* g *fills all tuples except one with identical values. One row gets a different value, which encodes a* 2*. Consider for example the tables depicted in Figure 4. Here,* g *filled the*

1	2
3	4

(a) d_1

5	6
7	8

(b) d_2

1	2	◯	5	6	□
3	4	◯	7	8	◯

(c) $g(d_1, d_2)$

Figure 4: An example for the two input databases d_1, d_2 of the algorithm g and its output $g(d_1, d_2)$. The Algorithm g adds an additional column to each database and encodes a 1 (equal symbols) into the additional column of first database and a 2 (one differing symbol) into the additional column of the second database.

additional column of d_1 *with* ◯*, while the symbols used in the additional column of* d_2 *are* □ *and* ◯*.*

With the information encoded in the additional columns, we can define a wrapper that applies the corresponding anonymization: Let $f(d_1', d_2')$ *be a wrapper function for* $(d_1', d_2') = g(d_1, d_2)$ *that calls* $f_1(d_i)$ *for the database* d_i' *with a* 1 *encoded and* $f_2(d_i)$ *for the database* d_i' *with an encoded* 2*, respectively. We anonymize* d_i *instead of* d_i' *as this database does not contain the column which was added by* g*. The mechanisms* f *and* g *provide a transformation from SC-IAD to SC-ID.*

Since f merely calls f_1 or f_2 and g adds an additional column to each of its input databases, the provided transformation from SC-IAD instances to SC-ID instances have an linear overhead in the size of the databases d_1 and d_2.

SC-IAD $\subseteq$ SC-IA: *Let d_1, d_2 be databases and f_1, f_2 database anonymization mechanisms of an SC-IAD instance. In order to transform a SC-IAD instance into an SC-IA instance, we need to specify an algorithm $h : DB \times DB \to DB$ and anonymization mechanisms f_1', f_2' with $f_1'(h(d_1, d_2)) = f_1(d_1)$ and $f_2'(h(d_1, d_2)) = f_2(d_2)$*

Consider an algorithm h that concatenates d_1 and d_2 tuple-wise and adds an additional column for every column in d_1 concatenated d_2. Into the first additional column, h encodes a 1 (see above) if the first column originates from d_1 and a 2 if the first column of the database is originally from d_2. In order to encode a 1, h chooses the same symbol for every entry in the additional column. In order to encode a 2, h chooses the same symbol for every entry in the additional column except for one entry where a different symbol is chosen. Consider for example Figure 5. Here, the first two tuples of the resulting database correspond to the first input database. Therefore, the first two additional columns only contain identical symbols, each.

1	2
3	4

(a) d_1

5	6
7	8

(b) d_2

1	2	5	6	◯	◯	□	□
3	4	7	8	◯	◯	◯	◯

(c) $h(d_1, d_2)$

Figure 5: An example for the two input databases d_1, d_2 of the algorithm h and its output $h(d_1, d_2)$. The algorithm h concatenates d_1 and d_2 tuple-wise and adds an additional column for each column. In each additional column, h encodes a 1 (identical symbols), if the corresponding column is from d_1 and a 2 (all but one symbols identical) if the corresponding column is from d_2.

With the information which column belongs to the first or the second database, we can define two algorithms, e. g. f_1' selects tuples belonging to the first database and anonymizes them with f_1. The mechanism f_i' discards each tuple in its input database, where the number in the corresponding column is different to i. Then it discards the extra columns and applies f_i to the result. Thus, f_1', f_2' and h allow a transformation from SC-IAD to SC-IA.

The algorithms f_i' execute a selection on the input database and then call f_1 or f_2. The resulting database of algorithm h contains all entries of the input databases and an additional column for each column of the input databases, such that the provided transformation from SC-IAD instances to SC-IA instances have an linear overhead in the size of the databases d_1 and d_2.

4.3 The Intersection Attack and Symbolic Databases

In Section 3.3, we have seen that ESVD and Locatability can be achieved in the symbolic framework. Therefore, the intersection attack (cf. Section 2.5, [GKS08]) can be performed. However, we stress that in general, Locatability is not even necessary for this attack, since it only decreases the size of the set of possible sensitive values.

In this section, we apply the intersection-algorithm to the example database we symbolized in Section 3.1. We use symbolic databases (cf. Figure 6) and we provide the adversary with some background knowledge in the form of a Locatability oracle. Instead of knowing the quasi identifiers of an individual (Alice) that occurs in both databases (as in a real scenario), we provide the adversary with the *symbolized* quasi identifiers of said individual.

a	b	e	f
a	b	e	g
a	b	e	h
a	b	e	h
a	c	e	i
a	c	e	g
a	c	e	h
a	c	e	h

(a) d_1

a	j	e	f
a	j	e	l
a	j	e	m
a	j	e	l
a	j	e	i
a	j	e	i
a	k	e	h
a	k	e	h

(b) d_2

Figure 6: Extended example of symbolic databases from Section 3.1. The first three columns denote the quasi identifiers while the last column denotes the sensitive value attribute. With these two databases, the Locatability oracle, and the symbolized quasi identifiers of an individual contained in both databases, an adversary is able to carry out the intersection attack.

The intersection attack can be carried out as follows: As a first step, the adversary uses the Locatability oracle in order to receive the buckets of the databases depicted in Figure 6 in which the sensitive value of Alice lies. The Locatability oracle for d_1 returns the set containing the sensitive values $\{f, g, h\}$, and the Locatability oracle for d_2 returns the set containing the sensitive values $\{f, l, m\}$, respectively. The adversary is able to determine that Alice has condition f by calculating the intersection of the two sets $\{f, g, h\} \cap \{f, l, m, \} = f$. This example shows that no background knowledge other than what is necessary to achieve Locatability is provided for a successful attack.

4.4 A General Attack on ESVD

In the previous section, we showed that the intersection attack can be carried out with symbolic databases. We now show that a generalization of the intersection attack with no side information at all can still lead to a disclosure of sensitive information.

Proposition 1 *Exploiting Exact Sensitive Value Disclosure is possible even without background knowledge about the attribute values.*

We now construct an example that serves as a witness for the symbolic non-composability of an anonymity notion that implements ESVD:

Proof 2 *Consider the database d depicted in Figure 7 and a mechanism f that partitions the database into buckets of size 2, calculates the product of each bucket with itself and*

a	1
b	2
c	3
d	4

Figure 7: An example database. It contains four tuples. $\{a, b, c, d\}$ are quasi-identifiers, $\{1, 2, 3, 4\}$ are sensitive attributes. Note that all entries are symbolic values. In particular, they cannot be added or multiplied. In our example, this database gets anonymized into the databases in Figures 8(a) and 8(b).

a	1
a	2
b	1
b	2
c	3
c	4
d	3
d	4

(a) Database d_1

a	1
a	3
c	1
c	3
b	2
b	4
d	3
d	4

(b) Database d_2

Figure 8: Two anonymized databases d_1 (a) and d_2 (b) that individually adhere 2-anonymity. The original database (cf. Figure 7) is anonymized by creating buckets of size 2 and joining each bucket with itself by its quasi-identifier (first attribute). The Database d_1 differs from d_2 because of different initial buckets in the original database. If d_1 and d_2 are intersected, the original database can be reconstructed.

joins the results. The results of f adhere 2-anonymity: Let the values in the first column be quasi-identifiers and the values in the second column be sensitive information. Then every quasi-identifier occurs at least two times in every database, making the databases 2-anonymous. Intersecting the databases in Figure 8, however, yields the original database of Figure 7, and intersecting symbolized versions of the anonymized databases yields a database structurally identical to the original database. This is possible due to the ESVD property of the anonymization mechanism used.

As shown by this example, ESVD can enable composition attacks on a structural level even if the adversary has no side information at all.

5 Summary and future work

In this paper, we presented a framework for studying the composition of anonymized database releases with no or very limited background knowledge. We provided a formal definition of composition notions and proved their equivalence. We provided strong evidence that Locatability end ESVD are the only necessary properties to enable composition

attacks by simulating them in our framework. We stress that all examples of composition attacks we gave are based solely on the structure of the databases and the Locatability Oracle we provided to the adversary, not on the meaning of its contents.

A possible extension of our framework is to model additional side information, e. g. relations of symbols. This is a natural extension, since values related in the real world are symbolized differently. We deem it significant to capture precisely the ratio of background information needed versus knowledge extracted from a release.

We also want to use the notion of symbolic databases in order to define anonymity notions more resilient to attacks involving background knowledge and composition. We propose to improve anonymity notions by including coarsening or aggregation of sensitive values [TZ11], similar to the idea of differential privacy. An important challenge for future work is to find tradeoffs between utility and privacy for methods that don't have undesirable properties like Locatability or Exact Sensitive Value Disclosure.

References

[BBG+11] Raghav Bhaskar, Abhishek Bhowmick, Vipul Goyal, Srivatsan Laxman, and Abhradeep Thakurta. Noiseless Database Privacy. In *ASIACRYPT*, pages 215–232, 2011.

[CRL07] Bee-Chung Chen, Raghu Ramakrishnan, and Kristen LeFevre. Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge. In *VLDB*, pages 770–781, 2007.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *EUROCRYPT*, pages 486–503, 2006.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC*, pages 265–284, 2006.

[DP91] G. Duncan and R. Pearson. Enhancing access to data while protecting confidentiality: prospects for the future. In *Statistical Science*, 1991.

[Dwo06] Cynthia Dwork. Differential Privacy. In *ICALP (2)*, pages 1–12, 2006.

[FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4), 2010.

[GKS08] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[GLP11] Johannes Gehrke, Edward Lui, and Rafael Pass. Towards Privacy for Social Networks: A Zero-Knowledge Based Definition of Privacy. In *TCC*, pages 432–449, 2011.

[JA05] Roberto J. Bayardo Jr. and Rakesh Agrawal. Data Privacy through Optimal k-Anonymization. In *ICDE*, pages 217–228, 2005.

[Kim86] J. Kim. A method for limiting disclosure of microdata based on random noise and transformation. In *Proceedings of the Section on Survey Research Methods for the American Statistical Association*, pages 370–374, 1986.

[KM12] Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.

[LDR05] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *SIGMOD Conference*, pages 49–60, 2005.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *ICDE*, pages 106–115, 2007.

[MGKV06] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, page 24, 2006.

[MKM+07] David J. Martin, Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke, and Joseph Y. Halpern. Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. In *ICDE*, pages 126–135, 2007.

[MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. Computational Differential Privacy. In *CRYPTO*, pages 126–142, 2009.

[MT07] Frank McSherry and Kunal Talwar. Mechanism Design via Differential Privacy. In *FOCS*, pages 94–103, 2007.

[MW04] Adam Meyerson and Ryan Williams. On the Complexity of Optimal K-Anonymity. In *PODS*, pages 223–228, 2004.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[PS86] M. Palley and J. Siminoff. Regression methodology based disclosure of a statistical database. In *Proceedings of the Section on Survey Research Methods for the American Statistical Association*, pages 382–387, 1986.

[SS98] Pierangela Samarati and Latanya Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and Its Enforcement through Generalization and Suppression. Technical report, CMU SRI, 1998.

[Swe02] Latanya Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10:557–570, October 2002.

[TZ11] Hongwei Tian and Weining Zhang. Extending *l*-diversity to generalize sensitive data. *Data Knowl. Eng.*, 70(1):101–126, 2011.

[WdW96] L. Willenborg and T. de Waal. *Statistical Disclosure Control in Practice*. Springer Verlag, 1996.

[WFWP07] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality Attack in Privacy Preserving Data Publishing. In *VLDB*, pages 543–554, 2007.

[XT06] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *SIGMOD Conference*, pages 229–240, 2006.

[ZYW05] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. Privacy-enhancing -anonymization of customer data. In *PODS*, pages 139–147, 2005.