J Comb Optim (2009) 18: 362–375
DOI 10.1007/s10878-009-9253-6

Fixed-parameter tractability of anonymizing data by suppressing entries

Patricia A. Evans · H. Todd Wareham · Rhonda Chaytor

Published online: 18 August 2009

Abstract A popular model for protecting privacy when person-specific data is released is *k-anonymity*. A dataset is k-anonymous if each record is identical to at least $(k-1)$ other records in the dataset. The basic k-anonymization problem, which minimizes the number of dataset entries that must be suppressed to achieve k-anonymity, is *NP*-hard and hence not solvable both quickly and optimally in general. We apply parameterized complexity analysis to explore algorithmic options for restricted versions of this problem that occur in practice. We present the first fixed-parameter algorithms for this problem and identify key techniques that can be applied to this and other k-anonymization problems.

Keywords Privacy · Anonymization · Parameterized complexity · Fixed-parameter tractability · Kernelization

1 Introduction

One of the most important promises a physician makes to a patient is that of confidentiality. Therefore, whether on paper or recorded electronically, patients expect their personal health information to remain confidential. At the same time, there is

P.A. Evans (✉)
Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada
e-mail: pevans@unb.ca

H.T. Wareham
Department of Computer Science, Memorial University, St. John's, NL, Canada
e-mail: harold@cs.mun.ca

R. Chaytor
School of Computing Science, Simon Fraser University, Vancouver, BC, Canada
e-mail: rhonda_chaytor@cs.sfu.ca

a conflicting need to release this information for health research. However, today in our growing digital society, guaranteeing patient privacy while providing researchers with worthwhile data has become increasingly difficult.

It is often desirable to make a dataset publicly available for research. In the past, it was believed that *de-identification*, i.e., removing obvious identifiers like social security number and name, would be sufficient in the protection of patient privacy. However, even if the individual pieces of information are common, combining them can be sufficient to either identify an individual uniquely or determine with high probability that he or she is part of a group with the same health information (Brankovic and Estivill-Castro 1999). The information used for this identification can be very general; for example, Sweeney showed how easily a de-identified medical record could be linked to an individual using little more then a zip code, date of birth, and gender—in particular, she pinpointed the governor of Massachusetts' medical record using publicly available medical records and a voter registry (six people in the state share his birth date, only three of which are male, and he is the only one of these three in his 5-digit ZIP code) (Sweeney 2002).

Work on statistical databases raises deeper concerns of the need to conceal individuals so that their information cannot be inferred with high probability. Information can be compromised even if the dataset can only be queried for basic statistics on ranges and multidimensional intervals; bounds on the confidential data in the interval can certainly be calculated based on the statistical results, and, with sufficient queries, the values of individual data items can be inferred either with certainty or with high probability (Brankovic and Estivill-Castro 1999). Many different techniques have been investigated to limit these risks and protect people's privacy. One method of risk limitation is to restrict the combinations of queries that can be asked, with the largest askable proportion of queries serving as the measure of the database's *usability*. Brankovic et al. investigated usability under Sum, Count, and Mean queries, finding that usability was inversely related to the number of unique rows (Brankovic et al. 1997), while, if Sum queries alone are used, all even-sized contiguous submatrices can be queried (Horak et al. 1999). Alternatively, noise can be added, enabling queries to be answered but compromising some of their accuracy, though some noise techniques can preserve important data patterns (Islam and Brankovic 2004). All privacy-protection techniques need to balance the need for high-quality statistics against the risks of revealing individual data.

Another model of privacy protection which limits the risk of re-identification of individuals whose data is stored in released datasets is k-Anonymity. A dataset is k-anonymous if each record is identical to at least $(k-1)$ others in the dataset (Samarati and Sweeney 1998; Sweeney 2002). A k-anonymous dataset limits re-identification and enables high usability of the remaining data with respect to statistical queries, so k-anonymization techniques seek to minimize information loss while transforming a collection of records to be k-anonymous. Information loss should be minimized to make k-anonymized datasets useful in subsequent analyses; moreover, k-anonymization must be done efficiently to make it an attractive option for privacy protection software systems.

There are many types of k-anonymization (see Chaytor 2006 for a survey), the most basic of which suppresses the smallest possible number of dataset entries to

achieve k-anonymity. More complex types address inferrability concerns, including requiring an identical group of public data to be linked to a variety of different medical results. Recently-derived complexity results (Aggarwal et al. 2005; Bonizzoni et al. 2007; Chaytor 2006; Meyerson and Williams 2004) imply that efficient algorithms for optimally or even approximately solving the general version of this problem (and hence many other types of k-anonymization) probably do not exist. This has motivated privacy researchers to concentrate on fast heuristic algorithms that produce good sub-optimal results (Chaytor 2007; Wang et al. 2004). However, given the need to minimize information loss, efficient optimal algorithms that solve restricted versions of k-anonymization that occur in practice would be preferable. Questions about the existence and derivation of such algorithms are best addressed using the theory of parameterized complexity (Downey and Fellows 1999). Parameterized complexity has previously been used to explore another problem based on anonymizing data, that of merging a limited number of rows or columns in a binary matrix to get a matrix of zeros (Fernau 2004); this problem is related to generalizing entries rather than completely suppressing them.

In this paper, we give the first parameterized complexity analysis of a k-anonymization problem, namely the entry suppression problem. This analysis includes algorithms which demonstrate the fixed-parameter tractability of this problem under a number of practically-useful restrictions; underlying these algorithms are three general frameworks that may be applicable to other k-anonymization problems. This paper is organized as follows: Sections 2 and 3 give background on k-anonymization by entry suppression and parameterized complexity theory, Sect. 4 gives parameterized hardness and tractability results for this problem, and Sect. 5 discusses these results and directions for future research.

2 Problem definition

Consider the following definitions, adapted from Meyerson and Williams (2004): Represent a database D of n entities described by m attributes as a row-set $X = \{x_1, x_2, \ldots, x_n\}$, where $x_i \in \Sigma^m$ for some attribute value-set Σ. Let $x_i[j]$ be the jth element of x_i. A function $f : D \to (\Sigma \cup \{*\})^m$ is a *suppressor* for D if:

$$\forall x_i \in D, \forall j \in \{1, 2, \ldots, m\}, \quad \{f(x_i[j]) \in \{x_i[j], *\}\}.$$

f thus transforms the database by replacing some entries by $*$, suppressing their original values. A transformed dataset $f(D)$ is *k-anonymous* if

$$\forall x_i \in D, \ \exists \text{ distinct } i_1, \ldots, i_{k-1} \in \{1, \ldots, n\} - \{i\}$$

such that

$$\forall 1 \leq j \leq k-1, \quad f(x_{i_j}) = f(x_i),$$

so that each row is identical to at least $k-1$ other rows. Our problem can now be stated as follows:

First	Last	Age	Town	Gender	Married
John	Smith	21	Fredericton	M	Yes
Joni	Smith	54	Fredericton	F	Yes
Sue	Taylor	37	Saint John	F	No
Teri	Walsh	37	Saint John	F	Yes

(a) Patient dataset

First	Last	Age	Town	Gender	Married
*	Smith	*	Fredericton	*	Yes
*	Smith	*	Fredericton	*	Yes
*	*	37	Saint John	F	*
*	*	37	Saint John	F	*

(b) 2-Anonymous patient dataset

Fig. 1 k-anonymization by dataset entry suppression

ENTRY SUPPRESSION (ESup)
Instance: An $n \times m$ dataset D over an alphabet Σ and integers $e, k \geq 0$.
Question: Can D be transformed into a k-anonymous dataset $f(D)$ by suppressing at most e entry values in D?

An example of k-anonymization by suppression is given in Fig. 1, in which the dataset in (a) is transformed into the 2-anonymous dataset in (b) by suppressing various entry values. This way of making the dataset anonymous allows the choice of entries to suppress to differ for different groups, suppressing them for individuals that they would distinguish while keeping them for others.

A number of complexity results have recently been derived for this problem. Initially, the ENTRY SUPPRESSION problem has been proven to be *NP*-hard when $k \geq 3$ (Meyerson and Williams 2004). Improving upon this result, the problem was shown to be *NP*-hard when $k \geq 3$ and $|\Sigma| \geq 3$ (Aggarwal et al. 2005), and this result has recently been improved to show that it is also *NP*-hard for binary alphabets ($|\Sigma| = 2$) with $k \geq 3$ (Bonizzoni et al. 2007).

Considering the approximability of the problem, the minimum number of suppressions can be approximated within $O(m \log k)$ in polynomial time (Meyerson and Williams 2004). It can also be approximated within $O(k)$ in polynomial time using a graph-based representation (Aggarwal et al. 2005). Considering inapproximability, polynomial-bounded absolute approximation or FPTAS algorithms do not exist for this problem unless $P = NP$ (Chaytor 2006). Furthermore, even if $k = 3$ and $|\Sigma| = 2$, the problem is *APX*-hard, so there are no polynomial time approximation schemes for it unless $P = NP$ (Bonizzoni et al. 2007).

3 Parameterized complexity analysis

Given intractability results such as those at the end of the previous section that rule out optimal efficient algorithms for general versions of a problem, one may still be in-

terested in algorithms whose non-polynomial running time is phrased purely in terms of an aspect x of that problem that is small in practice, e.g., $O(2^x n^3 + m^2)$. An *aspect* of the problem can be any characteristic of the problem input, often including the intended solution size. Parameterized complexity theory (Downey and Fellows 1999) directly addresses such questions by defining parameterized problems, which break instances into a parameter k and a main part n, and fixed-parameter tractability such that an algorithm's running time can only be non-polynomial in the parameter, i.e., the algorithm runs in $O(f(k)p(n))$ time, where f and p are arbitrary and polynomial functions, respectively. An algorithm with $O(f(k)p(n))$ running time can also be said to run in $O^*(f(k))$ time, eliminating the polynomial to focus on the exponential function of the parameter.

Denote a parameterized problem X with parameter k by $\langle k \rangle$-X. Two basic techniques for deriving fixed-parameter algorithms (Downey and Fellows 1999; Niedermeier 2006) are *bounded search* and *kernelization* (in which a classical exhaustive search and the candidate solution-set, respectively, are bounded by a function of the parameter). Fixed-parameter intractability can be shown via hardness for any of the classes of the W-hierarchy $= \{W[1], W[2], \ldots, W[P], \ldots, XP\}$, all of which seem (but have not been proven) to properly contain the class FPT of fixed-parameter tractable problems.

The analyses described above suffice in the case of problem restrictions encoded by single aspects; however, it is often of interest to look at restrictions operating over multiple aspects simultaneously. Such analyses can be simplified by the following relationships. For a problem of interest X, let S be a set of aspects for X, and we can consider any $S' \subseteq S$. Hardness is transferred from S' to subsets of S', and tractability is transferred from S to supersets of S', as follows:

- Given $S' \subseteq S'' \subseteq S$, if $\langle S' \rangle$-$X \in FPT$ then $\langle S'' \rangle$-$X \in FPT$.
- Given $S' \subseteq S'' \subseteq S$, if $\langle S'' \rangle$-X is C-hard for some parameterized complexity class C, then $\langle S' \rangle$-X is also C-hard.

4 The parameterized complexity of ENTRY SUPPRESSION

Several parameters of ENTRY SUPPRESSION that may be fixed or limited for different applications or sources of data are as follows:

m:	the number of columns, or fields, in the dataset
n:	the number of rows, or records, in the dataset
$\lvert\Sigma\rvert$:	the maximum number of different values in any column
k:	the desired minimum size of the anonymous groups
e:	the maximum number of entries that can be suppressed

We investigate the parameterized complexity of ENTRY SUPPRESSION relative to various subsets of $S = \{n, m, |\Sigma|, k, e\}$. The following relationships between these aspects will be exploited below:

- As the number of values in any column cannot exceed the number of rows, $|\Sigma| \leq n$.
- As the number of suppressed entries cannot exceed the number of dataset entries, $e \leq mn$.

– As the size of any identical group cannot exceed the number of rows, $k \leq n$.
– If the input dataset is not already k-anonymous, then at least one group must need suppressed entries, so at least k entries must be suppressed. Thus $k \leq e$ unless the input dataset is either k-anonymous or cannot be k-anonymized with only e suppressed entries.

The last relationship has more general consequences. Checking a dataset to determine if it is k-anonymous can be done by sorting and then grouping the rows, in $O(mn \log n)$ time, so $\langle e \rangle$-ENTRY SUPPRESSION reduces to $\langle e, k \rangle$-ENTRY SUPPRESSION. Since the reverse reduction also holds, using $\langle e \rangle$ as a parameter is equivalent to using $\langle e, k \rangle$ as parameters. This property also holds if additional parameters are included.

The alphabet size $|\Sigma|$ can vary greatly according to the dataset, from binary (such as gender) to highly multivalued (such as location). Tradeoffs are possible between the number of columns and the alphabet size; columns could be merged, with the values in the new column representing the combined result, or alternatively a column could be split. However, this type of alteration would restrict the types of suppressions possible, since merged entries would have to be kept or suppressed together rather than individually, and which results are optimal could change. The cost of suppressing a joint entry could also lead to adding cost weights for the columns.

Multivalued columns could instead be split into a set of binary columns, one for each of the original values, though this alteration would require multiple entries to be suppressed in order to obscure one value. This type of expansion into binary columns could be used to produce privacy through generalization rather than outright suppression, since, if only some of the possible types are suppressed, the value may be able to be categorized while still allowing anonymity.

4.1 Hardness results

Limiting the size of the anonymous group, k, and the size of the dataset's alphabet Σ does not, by themselves, enable parameterized solutions.

Theorem 4.1.1 $\langle k, |\Sigma| \rangle$-ESUP$\notin XP$ *unless* $P = NP$.

Proof For a decision problem Π with aspect-set parameter S, if Π is NP-hard when all aspects in S are constants then $\langle S \rangle$-Π is not in XP unless $P = NP$ (Wareham 1999). Since ESUP is NP-hard when $k \geq 3$ and $|\Sigma| \geq 3$ (Aggarwal et al. 2005), the result follows. □

These hardness results are also transferred to problem variants whose parameters are a subset of those proven hard. Thus $\langle k \rangle$-ESUP and $\langle |\Sigma| \rangle$-ESUP $\notin XP$ unless $P = NP$.

4.2 Fixed-parameter tractability results

While fixing k and $|\Sigma|$ is insufficient for fixed-parameter tractability, including either the number of rows or number of columns as a parameter does, in most cases,

1	A	B	C	D	E
2	A	B	C	D	E
3	A	B	C	D	E
4	X	B	C	D	E
5	X	B	C	D	E
6	A	B	Y	D	E
7	A	B	Y	D	E
8	A	B	C	D	Z
9	A	B	C	D	Z

(a) Original data set, with $\{1, 2, 3\}$ already 3-anonymous.

1	A	B	C	D	E
2	A	B	C	D	E
3	A	B	C	D	E
4	*	B	*	D	E
5	*	B	*	D	E
6	*	B	*	D	E
7	A	B	*	D	*
8	A	B	*	D	*
9	A	B	*	D	*

(b) 12 suppressions are needed to keep $\{1, 2, 3\}$ together.

3	*	B	C	D	E
4	*	B	C	D	E
5	*	B	C	D	E
2	A	B	*	D	E
6	A	B	*	D	E
7	A	B	*	D	E
1	A	B	C	D	*
8	A	B	C	D	*
9	A	B	C	D	*

(c) Only 9 suppressions are needed if $\{1, 2, 3\}$ are separated.

Fig. 2 Splitting up k-anonymous groups is needed to efficiently anonymize other rows

lead to fixed-parameter tractable algorithms. Note that limiting rows and columns simultaneously effectively fixes input size and hence is trivially in *FPT* (check all 2^{mn} entry-suppression combinations in $O(2^{mn}mn\log n)$ time). That being said, this basic exhaustive search algorithm forms part of the more complex algorithms given below.

Allowing the number of rows to be unlimited yields two cases, depending on whether or not all rows in the given dataset are distinct. The first case is fairly easy to deal with if m and $|\Sigma|$ form the parameter (as $n \leq |\Sigma|^m$, it can be solved in $O(2^{m|\Sigma|^m}nm\log n)$ time). Unfortunately, this will not typically hold, e.g., after all unique-value columns (such as name) have been removed as being identifying information. The second case, in which one has multiple copies of certain rows, initially seems to be potentially easier, since the dataset is already somewhat anonymous. However, identical rows are not necessarily grouped together, and even k identical rows may need to be split up in order to form groups with other similar rows. For example, we can have a dataset that includes k copies of one row x, and k groups of $k-1$ copies each of other rows r_i $(1 \leq i \leq k)$, where each row r_i differs from row x in column i only. If the k copies of row x are kept together, then all other groups (of size k or more) must be formed from rows that differ in at least two columns, thus requiring the suppression of at least $2k^2 - 2k$ entries. If instead the x rows are broken up, with one x row grouped with each group of r_i rows, then only one entry must be deleted from each row, suppressing only k^2 entries. This situation, illustrated for $k = 3$ in Fig. 2, shows that we cannot reduce the data that we need to consider by eliminating already k-anonymous rows. This example also illustrates some limitations on trying to kernelize datasets where the number of distinct rows is bounded.

To handle the situation where rows are not distinct, it suffices to add k to the previously-considered parameter. This is interesting in itself, because though k and $|\Sigma|$ do not by themselves yield fixed-parameter tractability, they do suffice when combined with m.

Theorem 4.2.1 $\langle m, |\Sigma|, k\rangle$-ESUP *can be solved in* $O(nm\log n + 2^{|\Sigma|^{2m}km}\cdot|\Sigma|^{2m}km\cdot(m\log|\Sigma| + \log k))$ *time, or* $O^*(2^{|\Sigma|^{2m}km}\cdot|\Sigma|^{2m})$.

Proof There will be at most $|\Sigma|^m$ distinct rows and, as discussed above, we cannot always group identical rows together. However, the number of groups needed will not be more than the number of distinct rows, since any row moved to another group must

Fig. 3 Algorithm 1. See Theorem 4.2.1 for details

```
Algorithm 1:
  sort the rows and group identical rows
  for each identical group i
      if the number of copies c_i > |Σ|^m · k
          remove these excess copies
          x_i = c_i − |Σ|^m · k
  for all combinations of entries in the remaining rows
      suppress the selected combination of entries
      sort the rows and group identical rows
      if all groups are of size ≤ k
          t = total number of suppressions of grouped rows
          for all initial groups i that have excess copies
              find the grouped copy that has the fewest suppressions s_i
              if x_i < k
                  t = t + x_i · s_i
          if t ≤ e
              return true
  return false
```

be needed in order to supplement a group of some other identical rows. Therefore the number of rows from any identical group that may be distributed to different groups is no more than $|\Sigma|^m \cdot k$. If an identical group has more row copies than this, the remaining rows can be included in whichever group (that already includes some copies of them) requires them to have the least number of suppressions per additional row. Since there are at most $|\Sigma|^m$ initial groups, where each group may give no more than $|\Sigma|^m \cdot k$ rows to other groups, we have a kernel of $|\Sigma|^{2m} \cdot k$ rows, each of length m.

Algorithm 1 (see Fig. 3) uses this principle to reduce and bound the number of rows need to be partitioned through trying each possible subset of entries to suppress. There will be at most $2^{|\Sigma|^{2m}km}$ suppressed-entry subsets to be considered, and each combination requires $|\Sigma|^{2m}k$ rows of length m to be sorted and grouped. Arriving at the total number of suppressions, including the rows that were not partitioned, is done for each original identical group by including the suppressions for those rows that were left out of the partition. Rows that are in groups of k or more should not be added in, since they are already part of a sufficiently large group. Thus Algorithm 1 runs in $O(nm \log n + 2^{|\Sigma|^{2m}km} \cdot |\Sigma|^{2m}km \cdot (m \log |\Sigma| + \log k))$ time, or $O^*(2^{|\Sigma|^{2m}km} \cdot |\Sigma|^{2m})$. □

Given the relationships between e and k mentioned earlier, the above also implies that $\langle m, |\Sigma|, e\rangle$-ESUP and $\langle m, |\Sigma|, e, k\rangle$-ESUP are in *FPT*.

If we allow the number of columns to be unlimited, we can instead limit the number of rows. Parameterizing by n supports two different fixed-parameter techniques, since the different partitions of the rows can be searched, and also the dataset can be kernelized by identifying columns that partition the rows in the same way.

Theorem 4.2.2 $\langle n\rangle$-ESUP *is solvable in* $O(nm \log m + 2^{n^{n+1}} n^{n+1} \log n)$ *time, or* $O^*(2^{n^{n+1}} n^{n+1})$, *through kernelization.*

```
Algorithm 2A:
  for all columns j from 1 to m
      relabel the entries of column j, starting from 1, in order
  sort the columns, and group them into sets of identical columns
  for each set j of identical columns
      remove all but one representative
      weight it by s_j, the number of columns in the identical group
  for all combinations of entries in the at most n × |Σ|^n table
      Sum = sum, over all selected entries, of their columns' weights
      if Sum ≤ e
          suppress all of these entries
          sort the table and group identical rows
          if all groups are of size ≥ k
              return true
  return false
```

Fig. 4 Algorithm 2A. See Theorem 4.2.2 for details

Proof We kernelize by reducing it to the related problem where the columns are weighted, and kernelizing the weighted problem by grouping columns. Unlike identical rows, which may need to be separated, identical columns can be grouped and handled together. If two columns produce an identical partitioning of the rows, then they need to have the same entries suppressed. If two identical columns have different entries suppressed, then they will partition the rows in different ways. If instead both columns have the same entries suppressed (choosing whichever had the fewest suppressed entries), then the partitioning produced by the other column will be removed from the overall partition (formed by overlaying the different column partitions). Removing a partitioning cannot decrease the size of the sets of rows in the overall partition, so making the columns identically suppressed will not increase the number of suppressions or break the k-anonymity condition. Thus the columns in the dataset can be restricted to the columns that produce different partitions of the rows, which is less than $|\Sigma|^n$. Each set of columns that produce the same partition can be represented by a single column, weighted by the size of the set, and there will be no more than $|\Sigma|^n$ such weighted columns. Algorithm 2A (see Fig. 4) implements this approach, and runs in $O(nm \log m + 2^{n \cdot |\Sigma|^n} \cdot |\Sigma|^n n \log n) = O(nm \log m + 2^{n^{n+1}} n^{n+1} \log n)$ time (as $|\Sigma| \leq n$). □

This algorithm's running time is also greatly improved if the alphabet size $|\Sigma|$ is fixed independently of n, so that $|\Sigma| \ll n$, since there will be at most $|\Sigma|^n$ columns left after the reduction. Algorithm 2A, then, is more useful for situations where $|\Sigma|$ is small.

Alternatively, the dataset can be anonymized by considering all possible partitions of the rows, and suppressing any column in a subset of the partition that is not identical in all the subset's rows. Any fast partition generating algorithm can be used, such as Er's (1988), which takes time $O(B_n)$, where B_n is the nth Bell number (counting the number of partitions of a set of n elements), a very fast growing progression generated by the recurrence

$$B_n = \sum_{k=0}^{n-1} B_k \binom{n-1}{k}$$

```
Algorithm 2B:
    for all partitions of the n rows
        if all groups are of size ≤ k
            for each column
                for each row
                    if the entry is different from other rows in the group
                        suppress the entry for all rows in that group
            if there are no more than e entries suppressed
                return true
    return false
```

Fig. 5 Algorithm 2B. See Theorem 4.2.3 for details

where $B_0 = 1$. B_n could also be used instead of $|\Sigma|^n$ for Algorithm 2A; the number of possible partitions based on a single column can be considered to be constrained either by the number of rows alone (using B_n) or by the alphabet size together with the number of rows (using $|\Sigma|^n$).

Theorem 4.2.3 $\langle n \rangle$-ESUP *is solvable in* $O(B_n \cdot n(m + \log n))$ *time, or* $O^*(B_n)$, *through bounded searching*.

Proof Instead of the kernelization approach of Algorithm 2A, Algorithm 2B (see Fig. 5) searches through all possible partitions into groups of size at least k, and suppresses whatever entries are needed to be suppressed in order for each group to be identical. There are B_n possible partitions, and the rows in any partition can be grouped in $O(n \log n)$ time through sorting and then have the necessary entries suppressed in $O(nm)$ time. Algorithm 2B thus solves the ENTRY SUPPRESSION problem in $O(B_n \cdot n(m + \log n))$ time. □

All of the algorithms above limit the number of symbols $|\Sigma|$, either directly or by limiting n. However, if we allow Σ to be arbitrarily large, we can still achieve fixed-parameter tractability by limiting m and e.

Theorem 4.2.4 $\langle m, e \rangle$-ESUP *is in FPT and can be solved using a combined kernelization and search strategy in* $O(2^{em}(e^3 2^e)^e mn \log n)$ *time, or* $O^*(2^{em}(e^3 2^e)^e)$.

Proof This result holds courtesy of Algorithm 3 (see Fig. 6). In order to be k-anonymizable, the number of rows in the original dataset that are in groups of less than k identical copies must be less than e, since each of these groups must have at least one entry suppressed per row. This produces a sub-dataset D'' of at most e rows, so there are at most 2^{em} different combinations of entries in that subset to consider suppressing and testing to determine if this suppression combination will work.

Once a suppression combination is to be tested, the other rows (already part of larger groups, D' in Algorithm 3) need to be considered to be added to the rows with suppressed entries. Since the suppression combination has determined which entries of D'' are suppressed, we consider rows from D' that are compatible with the groups of D''. We may need to consider more than just the excess rows of large groups (rows over the k minimum group size); however, we can only suppress e entries in total, so

```
Algorithm 3:
  sort and group identical rows
  select all rows in groups of size < k
  if more than e rows are selected
      return false
  D' = dataset—selected rows
  D'' = selected rows
  for all entry combinations in the selected rows D''
            where each row has at least one entry included
      suppress the combination of entries in D''
      e' = e—number of entries suppressed
      if e' ≥ 0
          sort and group identical rows from D''
          construct D''' from D' by:
              for all pairs i1, i2 of groups of identical rows in D'
                  let d(i1, i2) = Hamming distance between i1 and i2 rows
              sort the (i1, i2) pairs by nondecreasing d(i1, i2)
              D''' = empty
              for each pair i1, i2 of groups of identical rows in D', as sorted
                  size(i) = number of rows in i
                  m(i) = set of groups in D'' whose unsuppressed entries
                           match group i for all non-suppressed entries
                  c(i) = number of groups j already in D''' with
                           m(j) = m(i) and (size(j) = size(i) or size(j) ≥ e' + k)
                  if c(i1) < e'
                    move min(e', size(i1)) rows of group i1 from D' to D'''
                  if c(i2) < e'
                    move min(e', size(i2)) rows of group i2 from D' to D'''
          for each selection of e' rows from D'''
              for all combinations of up to e' entries in the selected rows
                  suppress that combination of entries
                  re-build D by uniting all rows in D', D'' and D'''
                  if at most e entries in D are suppressed
                     sort and group identical rows of D
                     if all groups of D are of size ≥ k
                        return true
  return false
```

Fig. 6 Algorithm 3. See Theorem 4.2.4 for details

we only need to consider at most e' rows from each group (where $e' = e$—the number of entries suppressed in the combination currently being tested).

Since $|\Sigma|$ is not bounded, there may be many groups in D' that are compatible with the groups of D''. We need, then, to find a boundary set D''' of rows from D' such that rows in $D'' - D'''$ are functionally equivalent to rows that are in D''', for the purpose of building a k-anonymous dataset. For example, if a group X of D'' that needs extending has one suppressed column, there may be $|\Sigma|$ different groups that agree with the non-suppressed columns of group X; they are functionally equivalent for extending group X. They may not, however, be functionally equivalent for extending other groups from D''; also, their sizes may differ, leading to a different number of rows being available or needing to be grouped after some rows have been removed to supplement X (or other groups). So, the boundary needs to include rows from groups that have different compatibilities with the groups in D'', and also are of different sizes. Since no more than e' rows can be taken from the boundary, groups of size

$\geq e' + k$ are considered to be functionally the same size, and indeed can be substituted for smaller groups.

Finally, after the groups in D'' are filled out with "donated" rows, the remainder of the rows still need to form groups of size at least k. The groups that donate rows to D''' thus need to be as close together as possible, so that the number of entries that need to be suppressed to merge them is minimized.

The rows of D''' are added to the groups of D'', put back into their original groups in D', or put together to form new groups; trying a suppression pattern of entries from e' rows selected from D''' will induce a complete grouping over all the rows. All of the rows are then reassembled and tested to determine if the suppression limit of e has been respected and the k-anonymity condition is met.

A minimum size set of suppressed entries is found by considering only those rows in D'' and D'''. Consider any minimum size set of suppressed entries that produces a k-anonymized dataset, and assume that it contains a row x that is not part of $D'' \cup D'''$. Let G_x refer to the k-anonymous group that row x is part of. Two cases need to be considered, depending on whether any rows from G_x are part of groups that use rows of D'''. If some row y from G_x is in D''', then there are at least e' rows z in D''' such that $d(y, z) \leq d(y, x)$; an identical group of up to k of them can be substituted for x (and any other rows from G_x not in D''') to produce a group with no more suppressed entries than G_x. If G_x itself does not contain any rows from D''', but other copies of one of its rows x' are found in a group that does contain rows from D''', the same argument can be applied to substitute for x' in $G_{x'}$, and, by extension, G_x, since D''' will contain enough closely related rows to be able to substitute for any cascade of groups related by common rows. If, on the other hand, no row from G_x or a related group is in D''', then x has a corresponding z row in D''' that is its functional equivalent and is from a group of D' that is the same size, so the rows identical to z can be substituted for the rows of x, with the rows close to z (from D''') in turn substituting for the rows close to x. Thus there is a minimum size set of suppressed entries that uses rows only from $D''' \cup D'''$.

With at most e rows in D'', there can be at most 2^e sets of rows $m(i)$ classified according to their functional compatibilities with groups of D''. Since D''' contains at most e rows from each of at most e groups of each of $e + 1$ possible sizes (sizes $k, k + 1, \ldots, e + k$) and 2^e functional types, the boundary set D''' will include at most $e^2(e + 1)2^e$ rows. Up to e rows from D''' need to be selected to extend D'', so all possible subsets of size at most e are potentially tried. Each subset selected needs to be tested for all possible e entries to be suppressed. These two steps can be combined so that up to e rows, with at most e entries suppressed, are chosen.

To test one of the $2^{e_1 m}$ possible suppression patterns (where e_1 is the number of rows in D''), then, requires testing each selection of no more than e rows from a set of $e^2(e+1)2^e$ rows, and trying each of no more than $2^{e_2 m}$ suppression patterns in the result (where $e_2 = e - e_1$). Considering these different suppression possibilities thus searched and tested, Algorithm 3 thus runs in $O(mn \log n + 2^{em} mn + 2^{em}(e^2(e + 1)2^e)^e mn \log n) = O(2^{em}(e^3 2^e)^e mn \log n)$ time. □

Table 1 Summary of Parameterized Results for ENTRY SUPPRESSION. For each *FPT* result, the numbers of the algorithms implying this result are given in parentheses

	–	k	e	k, e
–	*NP*-hard	$\notin XP$	???	???
$\|\Sigma\|$	$\notin XP$	$\notin XP$	???	???
m	???	???	$FPT(3)$	$FPT(3)$
n	$FPT(2)$	$FPT(2)$	$FPT(2)$	$FPT(2)$
$\|\Sigma\|, m$	???	$FPT(1)$	$FPT(1, 3)$	$FPT(1, 3)$
$\|\Sigma\|, n$	$FPT(2)$	$FPT(2)$	$FPT(2)$	$FPT(2)$

5 Conclusion

Table 1 summarizes the hardness and tractability results presented in this paper, including all results implied by pairwise parameter and parameter subset relationships. Note that as $k \leq e$, unless the dataset is already k-anonymous, the two rightmost columns are equivalent.

The algorithms underlying these *FPT* results illustrate techniques that can be applied to this problem, individually and together, and provide a general framework for algorithm development. Searching, either using a bounded search through the entire set or searching through a problem kernel, can consider all possible entry suppression combinations or all partitions. The results of such searches are equivalent, since partitioning the dataset into sets of rows will require any nonidentical columns in a group to have all their entries suppressed; similarly, an entry suppression combination will induce a partition of the rows into groups. In addition to the two search approaches, kernels can be found for problem variants where the number of rows or columns that differ or need to be considered can be bounded by a function of the parameters.

These algorithms show fixed-parameter tractability of the problem variants as indicated, and thus open up these variants to the investigation and development of faster algorithms that exploit those parameter combinations. The techniques presented here provide a framework for future algorithm development, especially with the general approach taken in algorithms such as Algorithm 3 that finds a core set of rows that need altering and then consider a limited set of other rows that are close to them.

The most pressing direction for future research is to improve the running times of the given *FPT* algorithms. Though most of these algorithms are not themselves practical, they do indicate which aspect-combinations allow fixed-parameter tractability. Past experience has shown that once a problem is shown to be in *FPT*, more sophisticated techniques can often be applied to derive practical algorithms (Niedermeier 2006), and similar work could potentially improve on the running times given here.

Also, the parameterized complexity of some variants relative to the parameters examined in this paper are still open. Of these, $\langle m, k \rangle$-ENTRY SUPPRESSION is of particular interest since the datasets to be anonymized frequently have small values of m and k. For example, according to current practices for statistical release adopted by the Newfoundland and Labrador Centre for Health Information (NLCHI), suitable values for m and k are 25 and 5, respectively (MacDonald 2005).

As well as improving the results within k-anonymity and the ENTRY SUPPRESSION problem, these results and problem variants can be expanded to incorporate other key issues that arise in database privacy, such as the need for an identical anonymous group to have different medical test results. Entry suppression can also be combined with some limited statistical querying, so that the entries may be suppressed but still have some statistical information about them available. Expanding suppression to value generalization should also be investigated.

Acknowledgements The research described in this paper was supported by an NSERC PGS-D scholarship (RC) and NSERC research grants 204923 (PE) and 228104 (TW).

References

Aggarwal G, Feder T, Kenthapadi K, Motwani R, Panigrahy R, Thomas D, Zhu A (2005) Approximation algorithms for k-anonymity. J Priv Technol, paper 20051120001

Brankovic L, Estivill-Castro V (1999) Privacy issues in knowledge discovery and data mining. In: Proceedings of Australian institute of computer ethics conference (AICEC99), pp 89–99

Brankovic L, Miller M, Horak P, Wrightson G (1997) Usability of compromise-free statistical databases. In: Proceedings of the ninth international conference on scientific and statistical database management (SSDBM 1997). IEEE Press, New York, pp 144–154

Bonizzoni P, Della Vedova G, Dondi R (2007) Anonymizing binary tables is APX-hard. The Computing Research Repository (CoRR) 0707.0421. http://arxiv.org/abs/0707.0421

Chaytor R (2006) Utility preserving k-anonymity. Technical report MUN-CS 2006-01, Dept Computer Science, Memorial University of Newfoundland

Chaytor R (2007) Allowing privacy protection algorithms to jump out of local optimums: an ordered greed framework. In: Bonchi F et al (eds) Proceedings of the 1st SIGKDD international workshop on privacy, security, and trust in KDD (PinKDD'07). LNCS, vol 4890. Springer, Berlin, pp 33–55

Downey R, Fellows M (1999) Parameterized complexity. Springer, Berlin

Er MC (1988) A fast algorithm for generating set partitions. Comput J 31:283–284

Fernau H (2004) Complexity of a {0, 1}-matrix problem. Australasian J Comb 29:273–300

Horak P, Brankovic L, Miller M (1999) A combinatorial problem in database security. Discrete Appl Math 91:119–126

Islam MZ, Brankovic L (2004) A framework for privacy preserving classification in data mining. In: Proceedings of the second workshop on Australasian information security, data mining and web intelligence, and software internationalisation (ACSW Frontiers 2004), pp 163–168

MacDonald (2005) personal communication

Meyerson A, Williams R (2004) On the complexity of optimal k-anonymity. In: Proceedings of 23rd ACM symposium on principles of database systems (PODS'04), pp 223–228

Niedermeier R (2006) Invitation to fixed-parameter algorithms. Oxford University Press, Oxford

Samarati P, Sweeney L (1998) Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report SRI-CSL-98-04, SRI International, Computer Science Laboratory

Sweeney L (2002) Achieving k-anonymity privacy protection using generalization and suppression. Int J Uncertain Fuzziness Knowl-Based Syst 10(5):571–588

Wang K, Yu P, Chakraborty S (2004) Bottom-up generalization: a data mining solution to privacy protection. In: Proceedings of 4th IEEE international conference on data mining (ICDM'04), pp 249–256

Wareham T (1999) Systematic parameterized complexity analysis in computational phonology. PhD thesis, Dept Computer Science, University of Victoria