CONFERENCE OF EUROPEAN STATISTICIANS

Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality Session 2

29-31 October 2019, the Hague, the Netherlands 16 September 2019

Algorithmic Matching Attacks on Optimally Suppressed Tabular Data

Kazuhiro Minami (Institute of Statistics, Japan)

kminami@ism.ac.jp

Abstract and Paper

The cell suppression problem (CSP) for tabular data has been studied in the community of official statistics for many years to prevent unintended disclosure of sensitive information from officially released data. In CSP, we determine the minimum set of suppressed cells that ensure the secrecy of sensitive cell values such that each sensitive cell possesses the feasibility interval of a sufficient width, which represents the range of possible values under the presence of linear relations concerning marginal sums. To compute the feasibility interval of a sensitive cell, previous algorithms for CSPs examine all possible combinations of suppressed cell values satisfying the marginal sum relations.

However, we find that a deterministic CSP algorithm is vulnerable to an adversary who knows that algorithm. Such an adversary can exclude some candidate cell values by applying the same CSP algorithm to the table whose suppressed cells are supplemented with those values. If the resulting suppression pattern does not match with that of the released suppressed table, the adversary concludes that that set of cell values cannot be part of the original table. Our experiments show that the actual widths of feasibility intervals in many tables are narrower than those assumed by a deterministic CSP algorithm.

Algorithmic Matching Attacks on Optimally Suppressed Tabular Data

Kazuhiro Minami[*,**,†] Yutaka Abe[**,†],

[*] The Institute of Statistical Mathematics, 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan; kminami@ism.ac.jp

[**] National Statistics Center, 19-1 Wakamatsu-cho, Shinjuku-Ku, Tokyo 162-8668, Japan; y-abe@ism.ac.jp

[†] The Graduate University for Advanced Studies (SOKENDAI), Shonan Village, Hayama, Kanagawa 240-0193, Japan

Abstract. This article is an extended abstract of our journal paper [16], which introduces the algorithmic matching attack on suppressed tabular data. The attack method narrows down the range of sensitive cell values by matching the suppression pattern of an original table with that of each candidate table. This matching attack is applicable only to tabular data that is suppressed with a pre-determined threshold value for the minimum width of the feasibility interval of a suppressed cell assuming a strong adversary with the knowledge of the suppression algorithm and security parameters. We also discuss possible solutions to the matching attack.

1 Introduction

The cell suppression problem (CSP) [6] has been studied for many years to protect sensitive information in tabular data properly. The goal of CSP is to determine a set of suppressed cells that ensure sufficient uncertainty on the values of sensitive cells under the presence of linear relations concerning marginal sums in the table.

Previous research on CSP [9] defines the safety of a suppressed table based on the notion of a *feasibility* interval. We represent a suppressed cell by a variable that takes an unknown value and examine the set of all possible values for that variable. When multiple variables are subject to linear constraints in CSP, the range of each variable becomes a linear segment between two end points. We call such a segment the feasibility interval of a cell variable and quantify the safety of the corresponding suppressed cell based on the *width* of the interval. Previous research [6] considers a suppressed cell safe if the width of its feasibility interval is greater than a given threshold.

However, we find that it is possible to exclude some feasible combinations of values of suppressed cell variables if we know that a *deterministic* CSP algorithm (e.g.,

[5, 8]) is used to suppress the original table. We can construct a candidate table of the original table by complementing its suppressed cells with a feasible combination of cell values. If we apply the same suppression algorithm to the candidate table, we are supposed to reproduce the same suppressed table. Otherwise, we conclude that that particular combination of cell values never coincide with the suppressed cell values of the original table. Since it is possible to enumerate all feasible combinations of suppressed cell values by solving indefinite linear equations of marginal constraints involving cell variables, we devise a systematic way of narrowing down the ranges of the feasibility intervals. We here assume an adversary who possesses the knowledge of the CSP algorithm.

We develop a matching attack for computing the *effective* feasibility intervals of suppressed cells in tabular data. The main idea is to compute the feasibility intervals of suppressed cells incrementally while performing the matching test on the suppression pattern of each candidate table. If a candidate table reproduces the suppression pattern of the original table, we consider that the cell values of that candidate table feasible and expand the ranges of their effective feasibility intervals accordingly. We evaluate the effectiveness of the proposed matching attack with a large number of synthetically generated frequency tables. Our experiments demonstrate that the matching attacks are both feasible and practical for tables of small and medium sizes compromising the safety of about 46% to 83% of the sensitive cell values.

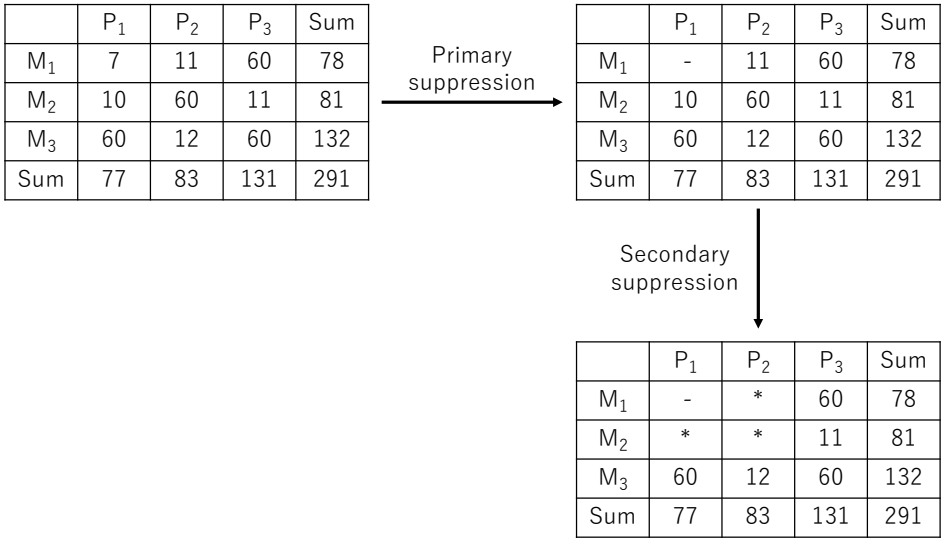
The rest of the paper is organized as follows. Section 2 describes the formulation of the CSP introducing the notion of the feasibility interval. Section 3 describes the matching attack that systematically examines all candidate tables and performs the matching test, and Section 4 demonstrates the effectiveness of the matching attacks experimentally. Section 5 covers related work and Section 6 discusses possible solutions to the matching attack. Section 7 finally concludes.

2 Background

In this section, we first formulate the cell suppression problem (CSP) introducing the notion of the feasibility interval. We next define the adversary model justifying our assumptions on an adversary's knowledge on the CSP algorithm.

2.1 Cell suppression problem (CSP)

We protect sensitive information in tabular data by suppressing the values of unsafe cells that are determined by applying some sensitivity rule [10] to each cell of the table. We call this process *primary* suppression and call suppressed cells in primary suppression *primary* suppressed cells accordingly. However, to perform primary suppression on a table is not sufficient to protect sensitive cell values because it is usually trivial to restore the original values by utilizing linear relationships among



	P_1	P_2	P_3	Sum
M_1	7	11	60	78
M_2	10	60	11	81
M_3	60	12	60	132
Sum	77	83	131	291

	P_1	P_2	P_3	Sum
M_1	-	11	60	78
M_2	10	60	11	81
M_3	60	12	60	132
Sum	77	83	131	291

	P_1	P_2	P_3	Sum
M_1	-	*	60	78
M_2	*	*	11	81
M_3	60	12	60	132
Sum	77	83	131	291

Figure 1: Two-step procedure for protecting sensitive cells in a frequency table. We consider the cell (M_1, P_1) sensitive because its value, 7, is smaller than the minimum frequency threshold, 10. We represent primary and secondary suppressed cells by symbols $-$ and $*$ respectively.

cell values concerning marginal sums. We, therefore, perform *secondary* suppression that additionally suppresses the values of non-sensitive cells to prevent an adversary from recomputing the cell values of primary suppressed cells. We call non-sensitive cells that are suppressed in secondary suppression *secondary* suppressed cells.

We illustrate this two-stage cell suppression procedure with an example of a frequency table in Figure 1. We first determine sensitive cells using the minimum frequency rule with a threshold value of 10. In this example, we determine that cell $(M_1,\ P_1,)$ is sensitive and suppress that cell value in the process of primary suppression. Since it is trivial to restore the suppressed value by considering a linear relationship either on the row sum or the column sum, we additionally suppress three non-sensitive cells $(M_1,\ P_2)$, $(M_2,\ P_1)$, and $(M_2,\ P_2)$. It is, however, not clear whether this particular choice of secondary suppressed cells adequately protects sensitive information in the primary suppressed cell. Also, we need to minimize information loss due to secondary suppression while protecting the value of the primary suppressed cell. We, therefore, define CSP at a high level as follows:

Definition 1 (Cell Suppression Problem (CSP)) *Given a secondary suppressed table T and a set S of primary suppressed cells in T, CSP determines a set S' of secondary suppressed cells with the minimum information loss under the constraint that the value of each primary suppressed cell $p \in S$ is properly protected.*

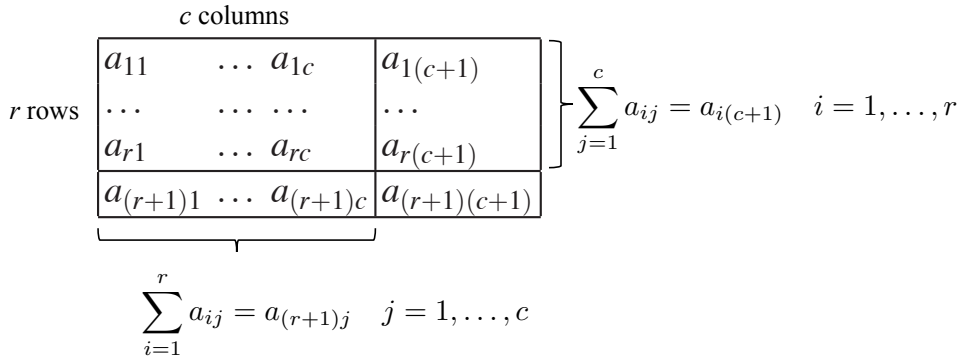


Figure 2: Linear relationships among cell values concerning marginal sums. We denote the value of a cell (i, j) by a_{ij}. Since there are r rows and c columns in the table, there are $r + c$ linear equations.

2.2 Safety of suppressed tabular data

To determine whether a given choice of secondary suppressed cells properly protects the values of primary suppressed cells, we need to verify that each primary suppressed cell has enough uncertainty on its value under the presence of linear relations concerning marginal sums. We consider a two-dimensional table T of r rows by c columns without loss of generality. Figure 2 shows linear relations in table T. We denote by a_{ij} the value of cell (i, j) in T and express the linear relations on marginal sums as follows:

$$\sum_{j=1}^{c} a_{ij} = a_{i(c+1)} \qquad \text{for every row } i, \text{ and} \tag{1}$$

$$\sum_{i=1}^{r} a_{ij} = a_{(r+1)c} \qquad \text{for every column } j. \tag{2}$$

We also assume that every cell takes a non-negative value as in the case of a frequency table, that is,

$$a_{ij} \geq 0 \quad \text{for every cell } (i, j) \tag{3}$$

Suppose that we suppress some cells in T and obtain a suppressed table T'. To quantify the uncertainty of suppressed cell values in T', we introduce a cell variable x_{ij} for each cell (i, j) of T'. We also introduce the notion of a suppression pattern y, which is a binary matrix of the same size as table T'. The suppression pattern y specifies which cells in table T' are suppressed such that:

$$y_{ij} = \begin{cases} 1 & \text{if cell } (i, j) \text{ in tabel } T' \text{ is suppressed} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We can obtain the lower bound $\underline{x_{ij}}$ of each primary suppressed cell (i, j) by solving the integer linear programming problem below.

$$\begin{aligned}
\text{minimize} \quad & x_{ij} && (5)\\
\text{subject to} \quad & \sum_{j=1}^{c} x_{ij} = x_{i(c+1)} \quad \text{for every row } i && (6)\\
& \sum_{i=1}^{r} x_{ij} = x_{(r+1)c} \quad \text{for every column } j && (7)\\
& x_{ij} = a_{ij} \quad \text{if} \quad \text{for every cell } (i,j) \text{ with } y_{ij} = 0 && (8)\\
& x_{ij} \geq 0 \quad \text{for every cell } (i,j) && (9)
\end{aligned}$$

Equation (8) sets the value for x_{ij} to be the actual cell value a_{ij} if that cell is suppressed in T', which is indicated by the value y_{ij} in suppression pattern y. Similarly, we can obtain the upper bound $\overline{x_{ij}}$ of cell (i, j) by computing the maximum value of x_{ij} under the same constraints (6)–(9) of the above integer linear programming problem.

We now define the feasibility interval of a suppressed cell in table T' as follows.

Definition 2 (Feasibility interval) *Given a suppressed table T' with a suppression pattern y, the feasibility interval for a suppressed cell (i, j) is an interval $[\underline{x_{ij}}, \overline{x_{ij}}]$ between the lower and upper bounds of cell variable x_{ij} and its width is defined as the delta between the two end points; that is,*

$$w_{ij} = \overline{x_{ij}} - \underline{x_{ij}}. \tag{10}$$

We quantify the uncertainty of a primary suppressed cell with the width of its feasibility interval and require that the width of the feasibility interval of every primary suppressed cell is greater than a given threshold value δ for the minimum width, as shown in Figure 3. We consider that a suppressed table T' is safe if every primary suppressed cell in T' is safe.

Definition 3 (Safety of a suppressed table T') *Given a suppressed table T' with a suppression pattern y, a set of primary suppressed cells P, and a minimum threshold δ, we say that a table T' is safe if, for every primary suppressed cell $(i, j) \in P$,*

$$w_{ij} > \delta \tag{11}$$

holds.

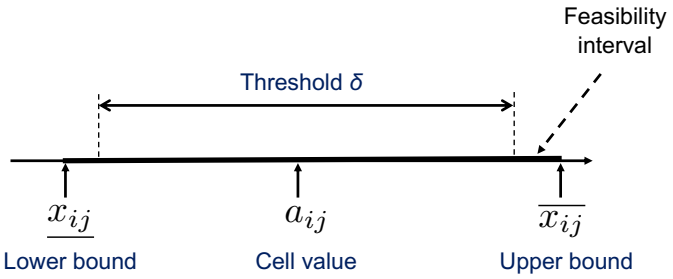


Figure 3: The safety condition on a primary suppressed cel at (i, j) based on the notion of its feasibility interval. The thick line represents the feasibility interval of a cell. A parameter δ is a threshold value for the minimum width of the feasibility interval.

2.3 Cell suppression problem (CSP)

The goal of solving a CSP for a given table is to determine the minimum set of cells to be suppressed that is sufficient to protect sensitive information in primary suppressed cells. We formulate CSP as an optimization problem of minimizing information loss under the safety condition in Definition 3 as follows.

Definition 4 (Cell suppression problem (CSP)) *Given a table T, a set of primary suppressed cells P, and a threshold δ for the minimum width of a feasibility interval, a CSP determines the suppression pattern y such that:*

$$\textit{minimize} \quad \sum_{i}^{r} \sum_{j}^{c} y_{ij} \tag{12}$$

$$\textit{subject to} \quad w_{ij} \geq \delta \quad \textit{for every primary suppressed cell } x_{ij} \in P, \tag{13}$$

where w_{ij} is the width of cell (i, j)'s feasibility interval in Definition 2.

For the simplicity of the presentation, we measure the information loss of a suppressed table by the number of suppressed cells as given in equation (12). Our matching attack in Section 3 is applicable to any goal function of CSP.

2.4 Adversary model

An adversary in this paper tries to infer sensitive cell values in a publicly released table. We assume that such an adversary knows the suppression algorithm for solving CSPs based on the Kerckhoffs' principle [18], which is one of the basic principles in cryptography. For example, τ-ARGUS [3] is publicly available with its source code and manual [7]. We also assume that an adversary knows security parameters, such as threshold values for the minimum frequency of cell values and the minimum width of a feasibility interval, because researchers who conduct data analysis on microdata

at a secure on-site facility (e.g., [12]) need to know output checking rules to get a permission from an output checker at a statistics office to bring their produced tables back home.

Even if a statistics agency keeps security parameters secret, an adversary can narrow down the ranges of those parameters by examining published suppressed tables, which are determined to be safe. The adversary can utilize the fact that a threshold for the minimum frequency must be smaller than any cell value of a published table and that a threshold for the minimum width of a feasibility interval must be smaller than the feasibility interval of any suppressed cell in published tables. Remember that the feasibility interval of each suppressed cell can be computed by solving the two linear programming problems in equations (5)–(9).

3 Matching attack

In this section, we describe the matching attack that eliminates a subset of candidate tables whose suppression patterns generated by the same CSP algorithm do not match with that of the original table. Note that our attack is applicable only to frequency tables in which a threshold for the minimum width of the feasibility interval of each sensitive cell is determined independently of cell values in a table.

3.1 Overview of the matching attack

We describe the matching attack on a secondary suppressed table produced by a *deterministic* CSP algorithm. Our attack exploits the fact that such a deterministic algorithm always outputs the same suppressed table from the same input table. For example, the optimal algorithm [8] and the network flow algorithm [5] in τ-ARGUS and the algorithm in SDCLink [14] are deterministic. The R package sdcTable [2] uses one of the CSP algorithms in τ-ARGUS underneath to solve a CSP. Note that if there exists a single optimal solution for a given CSP, any CSP algorithm for computing an optimal solution must find the same solution.

Figure 4 shows a conceptual scheme of the matching attack. An adversary first obtains the original suppressed table and enumerates every possible candidate table of the original table by complementing the suppressed cells of the suppressed original table with values that satisfy the constraints in (1), (2), and (3). The adversary next applies the CSP algorithm to each candidate table. We here assume that the adversary knows the CSP algorithm and its security parameters as described in Section 2.4. If a resulting suppressed table is the same as the suppressed original table, we consider that candidate table a *real* candidate; otherwise, we eliminate it from the list of candidate tables.

The main idea of the attack is to recompute the feasibility intervals of the suppressed cells in the suppressed original table by only considering the range of cell values in the matched candidate tables. We call the feasibility intervals recomputed

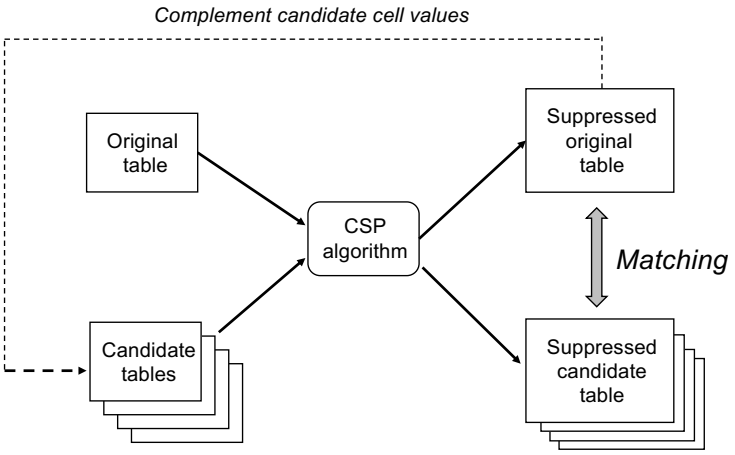


Figure 4: Conceptual scheme of the matching attack. The matching attack compares the original suppressed table with each candidate table generated by complementing values to suppressed cells in the original suppressed table.

with the real candidate tables the *effective* feasibility intervals. On the other hand, the feasibility interval in Definition 2 considers all candidate tables satisfying the linear and non-negativity constraints. Therefore, the width of the effective feasibility interval of a suppressed cell could be smaller than that of the feasibility interval in Definition 2 such that an adversary can infer that cell value more accurately than assumed by the CSP algorithm.

3.2 Enumerating candidate tables

We enumerate candidate tables by obtaining solutions of indefinite equations (1) and (2) involving cell variables of suppressed cells in the original table. We assume that cell variables for non-suppressed cells are replaced with the actual cell values as we do in Example 1 in Section 2.2. We denote the linear equations in (1) and (2) in a matrix form as follows.

$$Ax = b \tag{14}$$

where A is the coefficient matrix, x is a column vector of cell variables, and b is a column vector of marginal sums either on a row or a column. We here assume that we order cell variables in a table in a sequential order either by row or column.

Let $N(A)$ be the null space of matrix A such that

$$N(A) = \{y \in \mathcal{Z}^n \mid Ay = 0\} \tag{15}$$

where n is the number of cell variables and $\mathcal{Z}$ is the set of integers. If $x_1, x_2 \in \mathcal{Z}^n$ are solutions to $Ax = b$, that is $Ax_1 = b$ and $Ax_2 = b$, then $A(x_1 - x_2) = 0$. Thus,

the difference between any two solutions belongs to the null space $N(A)$. Therefore, any solution to the equation $Ax = b$ can be expressed as the sum of a fixed solution v and some element in $N(A)$; that is, the set S of all solutions to $Ax = b$ is defined as follows.

$$S = \{v + y \mid Av = b \wedge y \in N(A)\}. \tag{16}$$

Since the null space is represented by the linear combinations of independent vectors, we can enumerate all possible solutions that satisfy the non-negative constraints in (3) systematically.

Example 2 Suppose that we have a suppressed table in Table 1. There are four suppressed cells whose cell variables are v_1, v_2, v_3, and v_4, respectively.

Table 1: An example of a suppressed table. Each suppressed cell i is represented by a cell variable v_i.

	P_1	P_2	P_3	P_4	Sum
M_1	15	15	12	10	52
M_2	19	x_1	13	x_2	55
M_3	8	8	11	14	41
M_4	9	x_3	26	x_4	44
Sum	51	46	62	33	192

Then, the linear equations of the form $Ax = b$ in (14) are represented as

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 23 \\ 9 \\ 23 \\ 9. \end{bmatrix} \tag{17}$$

A vector $v = [14, 9, 9, 0]^T$ is a solution to equation (17) and the null space $N(A)$ is represented as

$$N(A) = \left\{ k \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T \mid \quad k \in \mathcal{Z} \right\} \tag{18}$$

Since each variable takes a non-negative value, we represent the set S of all possible combinations of cell values as

$$S = \left\{ \begin{bmatrix} 14 & 9 & 9 & 0 \end{bmatrix}^T + k \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^T \mid \quad k \in \mathcal{Z} \quad \wedge \quad 0 \leq k \leq 9 \right\} \tag{19}$$

In this example, the dimension of the null space is 1, but, in general, the null space is represented as a linear combination of multiple column vectors.

3.3 Algorithm for the matching attack

Our attack computes the effective feasibility interval of each suppressed cell by considering the cell values of candidate tables whose suppression pattern matches with that of the original table. We formally define the effective feasibility interval of a suppressed cell as follows.

Definition 5 (Effective feasibility interval) *Suppose that the function solveCSP for solving a CSP takes a table tbl and a list of security parameters parms as inputs and outputs a suppressed table stbl. Given a suppressed table T', the function solveCSP and its security parameters parm, the effective feasibility interval for a suppressed cell (i, j) is defined as $[\check{x_{ij}}, \hat{x_{ij}}]$ where*

$$\hat{x_{ij}} = \max\{x_{ij} \mid x \in S \ \wedge \ \forall(i,j) : x_{ij} \geq 0 \ \wedge \ solveCSP(T'(x), parms) = T'\} \quad (20)$$

$$\check{x_{ij}} = \min\{x_{ij} \mid x \in S \ \wedge \ \forall(i,j) : x_{ij} \geq 0 \ \wedge \ solveCSP(T'(x), parms) = T'\} \quad (21)$$

The width $\acute{w}_{ij}$ of the effective feasibility interval $[\check{x_{ij}}, \hat{x_{ij}}]$ is defined as

$$\acute{w}_{ij} = \hat{x_{ij}} - \check{x_{ij}}. \quad (22)$$

We denote S by the set of all solutions in (16) to $Ax = b$ in (14) and $T'(x)$ by the table whose suppressed cells in T' are complemented with the values of a cell vector x.

Algorithm 1 shows the pseudocode of the matching attack, which computes the effective feasibility intervals of suppressed cells in Definition 5. The algorithm takes a suppressed table *stbl*, a minimum frequency t, and a threshold width δ for the minimum width of the feasibility interval of a suppressed cell as inputs and outputs a list of actual feasibility intervals for the suppressed cells in the input table *stbl*.

Line 1 obtains a list of cell variables in a suppressed table *stbl*. Each variable v_i has an entry $vars[i]$, which maintains its minimum and maximum values. Lines 2–5 initialize each variable v_i's minimum and maximum values to ∞ and 0, respectively. Line 6 generates the linear equations of a matrix form (14) from the input suppressed table *stbl* by calling the function *genLinearEquations*, which produces the coefficient matrix A representing the linear equations regarding marginal row sums and column sums. Lines 7 obtains a fixed solution v to linear equations $Ax = b$ by calling the function $getFixedSolution$, which chooses the solution v closest to the origin of the coordinate system. We implement this function using the *lpSolve* package [1] for integer linear programming in the *R* language. Line 8 computes the null space $N(A)$ with the function *getNullSpace*. We represent the null space of A by the span of column vectors in $N(A)$. Line 9 enumerates all candidate solutions to $Ax = b$ where $x_i \geq 0$ for all i and put them into set S. The function *getAllSolutions* adds each linear combinations of column vectors in $N(A)$ into the fixed solution v and put into

Algorithm 1 Function $matchingAttack(T_0, s, \delta)$

Input: a suppressed table: $stbl$, a minimum frequency t, a threshold width: δ

Output: a set of actual feasibility intervals for the suppressed cells: $list$

```
1: vars ← getListOfVariables(stbl)
2: for i = 1 to length(vars) do
3:    vars[i].min ← ∞
4:    vars[i].max ← 0
5: end for
6: (A, b) ← genLinearEquations(stbl)
7: v ← getFixedSolution(A, b)
8: N(A) ← getNullSpace(A)
9: S ← getAllSolutions(v, N(A))
10: for all solution s ∈ S do
11:    tbl ← complement(stbl, s)
12:    stbl′ ← solveCSP(tbl, t, δ)
13:    if stbl′ = stbl then
14:       for each variable i in vars do
15:          if s[i] < vars[i].min then
16:             vars[i].min ← s[i]
17:          else if s[i] > vars[i].max then
18:             vars[i].max ← s[i]
19:          end if
20:       end for
21:    end if
22: end for
23: return vars
```

set S if all the components of the resulting solution take a non-negative value as specified in (16).

The for loop in lines 10–22 performs the matching test on every candidate table *tbl* and incrementally obtain the effective feasibility interval of each suppressed cell in *stbl*. Line 11 prepares a candidate table *tbl* by complementing suppressed cells in *stbl* with each solution s in the set S of all solutions. Line 12 performs cell suppression on a candidate table *tbl* with the function *solveCSP* for solving a CSP. As we discuss the adversary model in Section 2.4, we assume that an adversary uses the same deterministic algorithm *solveCSP* and security parameters t and δ as is used to produce an input suppressed table *stbl*. Line 13 checks whether a suppressed candidate table *stbl*$'$ is same as the input table *stbl*. If so, lines 14–20 update the feasibility intervals of each suppressed cell if the candidate cell values are outside the current ranges of the feasibility intervals. After examining all possible solutions for cell variables, line 23 returns a list of variables with their effective feasibility intervals.

4 Evaluation of the matching attacks

We evaluate the effectiveness of the matching attack in Section 3.1 experimentally using synthetically generated tables.

4.1 Setup

We implement the matching attack in Algorithm 1 in the R language. Also, to implement the function *solveCSP* in that algorithm, we use our implementation of the program for solving CSP based on the technique of Benders decomposition [4] in the R language [14]. We prepare synthetic two-dimensional frequency tables for the experiments as follows. We prepare 50 square tables of the same size in terms of the number of cells. We vary the number of cells, 16, 25, 36, and 49. We set the value of each cell in those tables to a randomly drawn from a Gaussian distribution with the average 15 and the standard deviation 10. We choose a threshold for the minimum frequency to be 5 and choose a threshold δ for the minimum width of the feasibility interval of every sensitive cell to be 8.

We perform cell suppression on each synthetic table T_i using the CSP algorithm and produce the secondary suppressed table T_i'. We next perform the matching attack in Algorithm 1 on T_i' with the same security parameters.

4.2 Safety of primary suppressed cells

We first evaluate how many primary suppressed cells in the test tables are unsafe; that is, the widths of their effective feasibility intervals are smaller than the minimum width of 8 in the case of our experiments. Table 2 shows the ratios of unsafe primary suppressed cells whose effective feasibility intervals become shorter than

Table 2: The ratios of unsafe primary suppressed cells attacked by the matching algorithm. We count the total sums of unsafe primary suppressed cells and primary suppressed cells in 50 different tables of each size.

#Cells in a table	#Unsafe primary suppressed cells	#Primary suppressed cells	Ratio of unsafe cells
16	48	104	0.46
25	117	170	0.69
36	190	230	0.83
49	226	271	0.83

Table 3: The ratios of unsafe cells by crossing the dimension of the null space and table size

	Dimension of the null space			
#Cells in a table	1	2	3	4
16	0.83	0.42	0.17	
25	1.00	0.71	0.57	0.83
36	1.00	0.87	0.78	0.82
49		0.81	0.89	0.73
Total	0.88	0.73	0.75	0.77

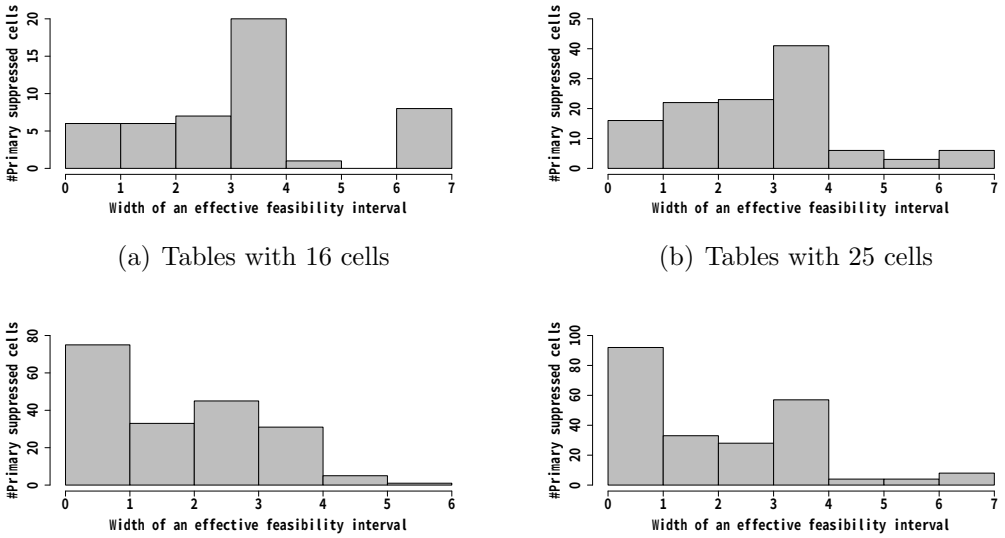
the threshold for the minimum width. The ratio for each table size is computed by dividing the sum of unsafe primary suppressed cells by the sum of all primary suppressed cells in 50 test tables of that size. The results show that about 46% to 83% of primary suppressed cells violates the safety requirement in Definition 3. We observe that as the size of a table becomes larger, the ratio of unsafe cells increases.

We next examine how the ratio of unsafe cells depends on the dimension of the null space $N(A)$ of the coefficient matrix A of a suppressed table in Section 3.2. Table 3 shows the ratios of unsafe cells by crossing tables with the dimensions of the null spaces and the number of cells in those tables. The results show that the lower the dimension of the null space is, the higher the risk of having unsafe cells in the table.

4.3 Distributions of the widths of the effective feasibility intervals of unsafe cells

We next examine how accurately we can infer the ranges of unsafe cells. Figure 5 shows the distributions of the widths of the effective feasibility intervals of unsafe primary suppressed cells in histograms. Surprisingly, we are able to determine the exact values of about 40% of the unsafe cells in tables of 36 cells and 49 cells. These results show that there are significant risks of revealing the exact values of sensitive

cells under the presence of the matching attack.



(a) Tables with 16 cells

(b) Tables with 25 cells

(c) Tables with 36 cells

(d) Tables with 49 cells

Figure 5: Distributions of the widths of effective feasibility intervals of primary suppressed cells.

5 Related work

To the best of our knowledge, there is no previous work on algorithmic attacks on optimally suppressed tabular data.

However, several researchers [19, 11] study the issue of algorithm-based attacks on anonymized microdata. The standard safety metrics for anonymized microdata is k-anonymity [17] and its variants (e.g., [13]). Wong et al [19] observes that most anonymization algorithms adopt the *minimality* principle to minimize information loss based on some criteria, and shows that sensitive information could be revealed from anonymized data based on the privacy metrics of l-diversity [13] if an adversary knows that the algorithm divides tuples into equivalence classes depending on the values of sensitive attributes. They also propose the primary metrics m-confidentiality that guarantees that an adversary who additionally possesses the knowledge on the anonymization algorithm cannot have confidence of more than $1/m$ on an individual's sensitive attribute value.

Although the concept of m-confidentiality is similar to safety conditions on feasibility intervals, which are extended to support continuous or ordered values, there is no efficient and systematic way of solving the inverse problem to verify the conditions of m-confidentiality. It is in general necessary to execute the anonymization

algorithm with all possible instances of input microdata. On the other hand, our contribution in this paper is to show experimentally that it is possible to eliminate a large portion of possible candidate tables that could be an input to the CSP algorithm by conducting the matching test.

Jin et al [11] establishes the requirements for algorithm-safe anonymization, which guarantees that the conditional probability on a given instance of original microdata does not change even if the knowledge on the anonymization algorithm is added to the conditional premise. Although they also identify conditions for algorithm-safe anonymization, that result is not directly applicable to algorithms for CSP.

6 Discussion

We discuss possible solutions to remedy the matching attack in this paper. To prevent the matching attack, we need to introduce some randomness to a CSP algorithm. There are several ways to introducing randomness at different levels of the algorithm. If the goal of a CSP is to minimize the number of suppressed cells as defined in equation (12) of Section 2.2, the CSP might have multiple solutions. If this is the case, a CSP algorithm can enumerate all optimal solutions and choose one of them randomly. However, it could be computationally expensive to compute all optimal solutions. Another possibility is to introduce randomness such that the algorithm chooses a different solution every time it is executed. For example, the optimal algorithm [8] computes an optimal solution by iteratively solving a master problem and a sub-problem while moving violated constraints from the latter to the former. Alternatively, we can randomize the optimal algorithm by randomly sorting the constraints to be added into the master problem.

If the objective of a CSP is to minimize the sum of the values of suppressed cells, that CSP is likely to have a single optimal solution. If this is the case, we need to sacrifice the optimality of a solution over the safety of tabular data. One possible way is to randomly choose a non-sensitive cell as a *dummy* sensitive cell and add it into a set of primary suppressed cells P for the modified CSP. However, when we randomize a CSP algorithm, we must be careful about the risks of differential attacks [15] on multiple versions of suppressed tables produced from the same original table.

7 Conclusion

In this paper, we describe the novel matching attack to infer sensitive cell values of a suppressed table exploiting the fact that the CSP algorithm, which is used to suppress a target table, is deterministic. The key idea is to eliminate candidate tables whose suppression patterns do not match with that of the target table. We experimentally demonstrate that the matching attack is successful to narrow down the ranges of a large portion of sensitive cell values to be smaller than the specified

minimum width of their feasibility intervals. We show that when the dimension of the null space of the coefficient matrix of a target table is low, there is significant risks of having unsafe cells whose values can be inferred accurately within a small range. As future work we plan to explore the possibility of developing a non-deterministic algorithm for CSP, which is resilient to the matching attack in this paper.

References

[1] lpsolve. https://cran.r-project.org/web/packages/lpSolve/index.html.

[2] sdctable: Methods for statistical disclosure control in tabular data. https://cran.r-project.org/web/packages/sdcTable/index.html.

[3] *tau*-argus homepage. http://research.cbs.nl/casc/tau.html.

[4] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, Dec 1962.

[5] Jordi Castro. Network flows heuristics for complementary cell suppression: An empirical evaluation and extensions. In *Inference Control in Statistical Databases, From Theory to Practice*, pages 59–73, London, UK, UK, 2002. Springer-Verlag.

[6] Jordi Castro. Recent advances in optimization techniques for statistical tabular data protection. *European Journal of Operational Research*, 216(2):257 – 269, 2012.

[7] Peter-Paul de Wolf, Anco Hundepool, Sarah Giessing, Juan-José Salazar, and Jordi Castro. *tau-ARGUS Version 4.1 User's Manual*. Statistics Netherlands, 2014.

[8] Matteo Fischétti and Jose Juan Salazar González. Models and algorithms for optimizing cell suppression in tabular data with linear constraints. *Journal of the American Statistical Association*, 95(451):916–928, 2000.

[9] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, and Peter-Paul de Wolf. *Statistical Control Disclosure*. Wiley, 2012.

[10] Anco Hundepool, Josep Domingo-Ferrer, Luisa Franconi, Sarah Giessing, Eric Schulte Nordholt, Keith Spicer, and Peter-Paul de Wolf. *Statistical Disclosure Control*. Wiley, 2012. 978-1119978152.

[11] Xin Jin, Nan Zhang, and Gautam Das. Algorithm-safe privacy-preserving data publishing. In *Proceedings of the 13th International Conference on Extending*

Database Technology, EDBT '10, pages 633–644, New York, NY, USA, 2010. ACM.

[12] Ryo Kikuchi and Kazuhiro Minami. On-site service and safe output checking in japan. In *Proceedings of the Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality*, 2017.

[13] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.

[14] Kazuhiro Minami and Yutaka Abe. Statistical disclosure control for tabular data in r. *Romanian Statistical Review*, (4):67–76, 2017.

[15] Kazuhiro Minami and Yutaka Abe. A first step towards statistical disclosure control on multiple linked tables. *Romanian Statistical Review*, (4):98–109, 2018.

[16] Kazuhiro Minami and Yutaka Abe. Algorithmic matching attacks on optimally suppressed tabular data. *Algorithms*, 12(8), 2019.

[17] P. Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, Nov 2001.

[18] Henk C. A. van Tilborg and Sushil Jajodia, editors. *Kerckhoffs' Law*, pages 675–675. Springer US, Boston, MA, 2011.

[19] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 543–554. VLDB Endowment, 2007.