Understanding DATABASE RECONSTRUCTION ATTACKS on Public Data

THESE ATTACKS ON STATISTICAL DATABASES ARE NO LONGER A THEORETICAL DANGER.

SIMSON GARFINKEL
JOHN M. ABOWD, AND
CHRISTIAN MARTINDALE
U.S. CENSUS BUREAU

In 2020 the U.S. Census Bureau will conduct the Constitutionally mandated decennial Census of Population and Housing. Because a census involves collecting large amounts of private data under the promise of confidentiality, traditionally statistics are published only at high levels of aggregation. Published statistical tables are vulnerable to DRAs (*database reconstruction attacks)*, in which the underlying microdata is recovered merely by finding a set of microdata that is consistent with the published statistical tabulations. A DRA can be performed by using the tables to create a set of mathematical constraints and then solving the resulting set of simultaneous equations. This article shows how such an attack can be addressed by adding noise to the published tabulations, so that the reconstruction no longer results in the original data. This has implications for the 2020 Census.

The goal of the census is to count every person once,

and only once, and in the correct place. The results are used to fulfill the Constitutional requirement to apportion the seats in the U.S. House of Representatives among the states according to their respective numbers.

In addition to this primary purpose of the decennial census, the U.S. Congress has mandated many other uses for the data. For example, the U.S. Department of Justice uses block-by-block counts by race for enforcing the Voting Rights Act. More generally, the results of the decennial census, combined with other data, are used to help distribute more than $675 billion in federal funds to states and local organizations.

Beyond collecting and distributing data on the American people, the Census Bureau is also charged with protecting the privacy and confidentiality of survey responses. All census publications must uphold the confidentiality standard specified by Title 13, Section 9 of the U.S. Code, which states that Census Bureau publications are prohibited from identifying "the data furnished by any particular establishment or individual." This section prohibits the Census Bureau from publishing respondents' names, addresses, or any other information that might identify a specific person or establishment.

Upholding this confidentiality requirement frequently poses a challenge, because many statistics can inadvertently provide information in a way that can be attributed to a particular entity. For example, if a statistical agency *accurately* reports that there are two persons living on a block and that the average age of the block's residents is 35, that would constitute an improper disclosure of personal information, because

one of the residents could look up the data, subtract their contribution, and infer the age of the other.

Of course, this is an extremely simple example. Statistical agencies have understood the risk of such unintended disclosure for decades and have developed a variety of techniques to protect data confidentiality while still publishing useful statistics. These techniques include *cell suppression*, which prohibits publishing statistical summaries from small groups of respondents; *top-coding*, in which ages higher than a certain limit are coded as that limit before statistics are computed; *noise-injection*, in which random values are added to some attributes; and *swapping*, in which some of the attributes of records representing different individuals or families are swapped. Together, these techniques are called SDL (statistical disclosure limitation).

Computer scientists started exploring the issue of statistical privacy in the 1970s with the increased availability of interactive query systems. The goal was to build a system that would allow users to make queries that would produce summary statistics without revealing information about individual records. Three approaches emerged: auditing database queries, so that users would be prevented from issuing queries that zeroed in on data from specific individuals; adding noise to the data stored within the database; and adding noise to query results.[1] Of these three, the approach of adding noise proved to be the easiest, because the complexity of auditing queries increased exponentially over time—and, in fact, was eventually shown to be NP (nondeterministic polynomial)-hard.[8] Although these results were all couched in the

language of interactive query systems, they apply equally well to the activities of statistical agencies, with the *database* being the set of confidential survey responses, and the *queries* being the schedule of statistical tables that the agency intends to publish.

In 2003, Irit Dinur and Kobbi Nissim showed that it isn't even necessary for an attacker to construct queries on a database carefully to reveal its underlying confidential data.[4] Even a surprisingly small number of random queries can reveal confidential data, because the results of the queries can be combined and then used to "reconstruct" the underlying confidential data. Adding noise to either the database or to the results of the queries decreases the accuracy of the reconstruction, but it also decreases the accuracy of the queries. The challenge is to add sufficient noise in such a way that each individual's privacy is protected, but not so much noise that the utility of the database is ruined.

Subsequent publications[3,6] refined the idea of adding noise to published tables to protect the privacy of the individuals in the data set. Then in 2006, Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith proposed a formal framework for understanding these results. Their paper, "Calibrating Noise to Sensitivity in Private Data Analysis,"[5] introduced the concept of *differential privacy*. They provided a mathematical definition of the privacy loss that persons suffer as a result of a data publication, and they proposed a mechanism for determining how much noise needs to be added for any given level of privacy protection. (The authors were awarded the Test of Time award at the Theory of Cryptography Conference in 2016

and the Godel Prize in 2017.)

The 2020 census is expected to count roughly 330 million people living on roughly 8.5 million blocks, with some inhabited blocks having as few as a single person and other blocks having thousands. With this level of scale and diversity, it is difficult to visualize how such a data release might be susceptible to database reconstruction. We now know, however, that reconstruction would in fact pose a significant threat to the confidentiality of the 2020 microdata that underlies unprotected statistical tables if privacy-protecting measures are not implemented. To help understand the importance of adopting formal privacy methods, this article presents a database reconstruction of a much smaller statistical publication: a hypothetical block containing seven people distributed over two households. (The 2010 U.S. Census contained 1,539,183 census blocks in the 50 states and the District of Columbia with between one and seven residents. The data can be downloaded from https://www2.census.gov/census_2010/01-Redistricting_File--PL_94-171/.)

Even a relatively small number of constraints results in an exact solution for the blocks' inhabitants. Differential privacy can protect the published data by creating uncertainty. Although readers may think that the reconstruction of a block with just seven people is an insignificant risk for the country as a whole, this attack can be performed for virtually every block in the United States using the data provided in the 2010 census. The final section of this article discusses the implications of this for the 2020 decennial census.

AN EXAMPLE DATABASE RECONSTRUCTION ATTACK

To present the attack, let's consider the census of a fictional geographic frame (for example, a suburban block), conducted by a fictional statistical agency. For every block, the agency collects each resident's age, sex, and race, and publishes a variety of statistics. To simplify the example, this fictional world has only two races—black or African American, and white—and two sexes—female and male.

The statistical agency is prohibited from publishing the raw microdata and instead publishes a tabular report. Table 1 shows fictional statistical data for a fictional block

TABLE 1: **FICTIONAL STATISTICAL DATA FOR A FICTIONAL BLOCK**

		AGE		
STATISTIC	**GROUP**	**COUNT**	**MEDIAN**	**MEAN**
1A	total population	7	30	38
2A	female	4	30	33.5
2B	male	3	30	44
2C	black or African American	4	51	48.5
2D	white	3	24	24
3A	single adults	(D)	(D)	(D)
3B	married adults	4	51	54
4A	black or African American female	3	36	36.7
4B	black or African American male	(D)	(D)	(D)
4C	white male	(D)	(D)	(D)
4D	white female	(D)	(D)	(D)
5A	persons under 5 years	(D)	(D)	(D)
5B	persons under 18 years	(D)	(D)	(D)
5C	persons 64 years or over	(D)	(D)	(D)

Note: Married persons must be 15 or over

published by the fictional statistics agency. The "statistic" column is for identification purposes only.

Notice that a substantial amount of information in table 1 has been suppressed—marked with a (D). In this case, the statistical agency's disclosure-avoidance rules prohibit it from publishing statistics based on one or two people. This suppression rule is sometimes called "the rule of three," because cells in the report sourced from fewer than three people are suppressed. In addition, complementary suppression has been applied to prevent subtraction attacks on the small cells.

Encoding the constraints

The database can be reconstructed by treating the attributes of the persons living on the block as a collection of variables. A set of constraints is then extracted from the published table. The database reconstruction finds a set of attributes that are consistent with the constraints. If the statistics are highly constraining, then there will be a single possible reconstruction, and the reconstructed microdata will necessarily be the same as the underlying microdata used to create the original statistical publication. Note that there must be at least one solution because the table is known to be tabulated from a real database.

For example, statistic 2B states that three males live in the geography. This fictional statistical agency has previously published technical specifications that its computers internally represent each person's age as an integer. The oldest verified age of any human being was 122.[14] If we allow for unreported supercentenarians and consider 125 to the oldest possible age of a human

being, there are only a finite number of possible age combinations, specifically:

$$\binom{125}{3} = \frac{125 \times 124 \times 123}{3 \times 2 \times 1} = 317{,}750$$

Within the 317,750 possible age combinations, however, there are only 30 combinations that satisfy the constraints of having a median of 30 and a mean of 44 (see Table 1). (Notice that the table does not depend on the oldest possible age, so long as it is 101 or over.) Applying the constraints imposed by the published statistical tables, the possible combinations of ages for the three males can be reduced from 317,750 to 30. Table 2 shows the 30 possible ages for which the median is 30 and the mean is 44

TABLE 2: **POSSIBLE AGES FOR A MEDIAN OF 30 AND MEAN OF 44**

A	B	C	A	B	C	A	B	C
1	30	101	11	30	91	21	30	81
2	30	100	12	30	90	22	30	80
3	30	99	13	30	89	23	30	79
4	30	98	14	30	88	24	30	78
5	30	97	15	30	87	25	30	77
6	30	96	16	30	86	26	30	76
7	30	95	17	30	85	27	30	75
8	30	94	18	30	84	28	30	74
9	30	93	19	30	83	29	30	73
10	30	92	20	30	82	30	30	72

To mount a full reconstruction attack, an attacker extracts all of these constraints and then creates a single mathematical model embodying all constraints. An automated solver can then find an assignment of the variables that satisfies these constraints.

To continue with the example, statistic 1A establishes the universe of the constraint system. Because the block contains seven people, and each has four attributes (age, sex, race, and marital status), that creates 28 variables, representing those four attributes for each person. These variables are A1... A7 (age), S1... S7 (sex), R1... R7 (race), and M1... M7 (marital status), as shown in table 3. The table

TABLE 3: **VARIABLES ASSOCIATED WITH THE RECONSTRUCTION ATTACK.**

PERSON	AGE	SEX	RACE	MARITAL STATUS
1	A1	S1	R1	M1
2	A2	S2	R2	M2
3	A3	S3	R3	M3
4	A4	S4	R4	M4
5	A5	S5	R5	M5
6	A6	S6	R6	M6
7	A7	S7	R7	M7
KEY				
female		0		
male		1		
black or African American			0	
white			1	
single				0
married				1

shows the variables associated with the DRA. The coding for the categorical attributes is presented in the key.

Because the mean age is 38, we know that:

$$A1 + A2 + A3 + A4 + A5 + A6 + A7 = 7 \times 38$$

The language Sugar[13] is used to encode the constraints in a form that can be processed by a SAT (satisfiability) solver. Sugar represents constraints as s-expressions.[11] For example, the age combination equation can be represented as:

```
; First define the integer variables,
; with the range 0..125
(int A1 0 125)
(int A2 0 125)
(int A3 0 125)
(int A4 0 125)
(int A5 0 125)
(int A6 0 125)
(int A7 0 125)

; Statistic 1A: Mean age is 38
(= (+ A1 A2 A3 A4 A5 A6 A7)
    (* 7 38)
)
```

Once the constraints in the statistical table are turned into s-expressions, Sugar solves them with a brute-force algorithm. Essentially, Sugar explores every possible combination of the variables, until a combination is

found that satisfies the constraints. Using a variety of heuristics, SAT solvers are able to rapidly eliminate many combinations of variable assignments.

Despite their heuristic complexity, SAT solvers can process only those systems that have Boolean variables, so Sugar transforms the s-expressions into a much larger set of Boolean constraints. For example, each age variable is encoded using unary notation as 126 Boolean variables. Using this notation, the decimal value 0 is encoded as 126 false Boolean variables, the decimal value 1 is encoded as 1 true and 125 false values, and so on. Although this conversion is not space efficient, it is fast, provided that the integers have a limited range.

To encode the median age constraint, the median of a group of numbers is precisely defined as the value of the middle number when the numbers are arranged in sorted order (for the case in which there is an odd number of numbers). Until now, persons 1 through 7 have not been distinguished in any way: the number labels are purely arbitrary. To make it easier to describe the median constraints, we can assert that the labels must be assigned in order of age. This is done by introducing five constraints, which has the side effect of eliminating duplicate answers that have simply swapped records, an approach called *breaking symmetry*:[12]

```
(<= A1 A2)
(<= A2 A3)
(<= A3 A4)
(<= A4 A5)
(<= A6 A7)
```

Having asserted that the labels are in chronological order, we can constrain the age of the person in the middle to be the median:

```
(= A4 30)
```

This code fragment assures that the output is sorted by age. This technique also does a good job of eliminating duplicate answers that have swapped records.

Sugar has an "`if`" function that allows encoding constraints for a subset of the population. Recall that statistic 2B contains three constraints: there are three males, their median age is 30, and their average age is 44. The value 0 represents a female, and 1 represents a male:

```
#define FEMALE 0
#define MALE 1
```

Using the variable `Sn` to represent the sex of person `n`, we then have the constraint:

```
S1 + S2 + S3 + S4 + S5 + S6 + S7 = 3
```

This can be represented as:

```
(= (+ S1 S2 S3 S4 S5 S6 S7) 3)
```

Now, using the `if` function, it is straightforward to create a constraint for the mean age 44 of male persons:

```
(= (+ (if (= S1 MALE) A1 0) ; average male age
      (if (= S2 MALE) A2 0)
      (if (= S3 MALE) A3 0)
      (if (= S4 MALE) A4 0)
      (if (= S5 MALE) A5 0)
      (if (= S6 MALE) A6 0)
      (if (= S7 MALE) A7 0)
      )
   (* 3 44))
```

Table 1 translates into 164 individual s-expressions extending over 457 lines. Sugar then translates this into a single Boolean formula consisting of 6,755 variables arranged in 252,575 clauses. This format is called the CNF (conjunctive normal form) because it consists of many clauses that are combined using the Boolean AND operation.

Interestingly, we can even create constraints for the suppressed data. Statistic 3A is suppressed, so we know that there are 0, 1, or 2 single adults, assuming that no complementary suppression was required. Let Mn represent the marital status of person n:

```
#define SINGLE 0
#define MARRIED 1

(int SINGLE_ADULT_COUNT 0 2)
(= (+ (if (and (= M1 SINGLE) (> A1 17)) 1 0)
      (if (and (= M2 SINGLE) (> A2 17)) 1 0)
      (if (and (= M3 SINGLE) (> A3 17)) 1 0)
      (if (and (= M4 SINGLE) (> A4 17)) 1 0)
```

```
      (if (and (= M5 SINGLE) (> A5 17)) 1 0)
      (if (and (= M6 SINGLE) (> A6 17)) 1 0)
      (if (and (= M7 SINGLE) (> A7 17)) 1 0))
 SINGLE_ADULT_COUNT)

(>= SINGLE_ADULT_COUNT 0)
(<= SINGLE_ADULT_COUNT 2)
```

Translating the constraints into CNF allows them to be solved using any solver that can solve NP-complete program, such as a SAT solver, an SMT (satisfiability module theories) solver, or MIP (mixed integer programming) solver. There are many such solvers, and most take input in the so-called DIMACS file format, which is a standardized form for representing CNF equations. The DIMACS format (named for the Center for Discrete Mathematics and Theoretical Computer Science) was popularized by a series of annual SAT solver competitions. One of the results of these competitions was a tremendous speed-up of SAT solvers over the past two decades. Many solvers can now solve CNF systems with millions of variables and clauses in just a few minutes, although some problems do take much longer. Marijn Heule and Oliver Kullmann discussed the rapid advancement and use of SAT solvers in their 2017 article, "The Science of Brute Force."[7]

The open-source PicoSAT[2] solver is able to find a solution to the CNF problem detailed here in roughly two seconds on a 2013 MacBook Pro with a 2.8-GHz Intel i7 processor and 16 GB of RAM (although the program is not limited by RAM), while the open-source Glucose SAT solver can solve the problem in under 0.1 seconds on the same

computer. The stark difference between the two solvers shows the speed-up possible with an improved solving algorithm.

Exploring the solution universe

PicoSAT creates a satisfying assignment for the 6,755 Boolean variables. After the solver runs, Sugar can translate these assignments back into integer values of the constructed variables. (SMT and MIP solvers can represent the constraints at a higher level of abstraction, but for our purposes a SAT solver is sufficient.)

There exists a solution universe of all the possible solutions to this set of constraints. If the solution universe contains a single possible solution, then the published statistics completely reveal the underlying confidential data—provided that noise was not added to either the microdata or the tabulations as a disclosure-avoidance mechanism. If there are multiple satisfying solutions, then any element (person) in common among all of the solutions is revealed. If the equations have no solution, either the set of published statistics is inconsistent with the fictional statistical agency's claim that it is tabulated from a real confidential database or an error was made in that tabulation. This doesn't mean that a high-quality reconstruction is not possible. Instead of using the published statistics as a set of constraints, they can be used as inputs to a multidimensional objective function: the system can then be solved using another kind of solver called an optimizer.

Normally SAT, SMT, and MIP solvers will stop when they find a single satisfying solution. One of the advantages

of PicoSAT is that it can produce the solution universe of all possible solutions to the CNF problem. In this case, however, there is a single satisfying assignment that produces the statistics in table 1. That assignment is seen in Table 4.

Table 1 provides some redundant constraints on the solution universe: some of the constraints can be dropped while preserving a unique solution. For example, dropping statistic 2A, 2B, 2C, or 2D still yields a single solution, but dropping 2A *and* 2B increases the solution universe to eight satisfying solutions. All of these solutions contain the reconstructed microdata records 8FBS, 36FBM, 66FBM, and 84MBM. This means that even if statistics 2A and 2B are suppressed, we can still infer that these four microdata records must be present.

Statistical agencies have long used suppression in an attempt to provide privacy to those whose attributes are present in the microdata, although the statistics that they typically drop are those that are based on a small number

TABLE 4: **A SINGLE SATISFYING ASSIGNMENT**

AGE	SEX	RACE	MARITAL STATUS		SOLUTION #1
8	F	B	S		8FBS
18	M	W	S		18MWS
24	F	W	S	=	24FWS
30	M	W	M		30MWM
36	F	B	M		36FBM
66	F	B	M		66FBM
84	M	B	M		84MBM

of persons. How effective is this approach?

In table 1, statistic 4A is an obvious candidate for suppression—especially given that statistics 4B, 4C, and 4D have already been suppressed to avoid an inappropriate statistical disclosure.

Removing the constraints for statistic 4A increases the number of solutions from one to two, shown in table 5.

DEFENDING AGAINST A DRA

There are three approaches for defending against a database reconstruction attack. The first is to publish less statistical data—this is the approach taken by legacy disclosure-avoidance techniques (cell suppression, top-coding, and generalization). The second and third approaches involve adding noise, or randomness. Noise can be added to the statistical data being tabulated or to the results after tabulation. Each approach is considered here.

Option 1: Publish less data

Although it might seem that publishing less statistical data is a reasonable defense against the DRA, this choice

TABLE 5: **SOLUTIONS WITHOUT STATISTIC 4A**

SOLUTION #1	SOLUTION #2
8FBS	2FBS
18MWS	12MWS
24FWS	24FWM
30MWM	30MBM
36FBM	36FWS
66FBM	72FBM
84MBM	90MBM

may severely limit the number of tabulations that can be published. A related problem is that, with even a moderately small population, it may be computationally infeasible to determine when the published statistics still identify a sizable fraction of individuals in the population.

Option 2: Apply noise before tabulation

This approach is called *input noise injection*. For example, each respondent's age might be randomly altered by a small amount. Input noise injection doesn't prevent finding a set of microdata that is consistent with the published statistics, what we call *database reconstruction*, but it limits the value of the reconstructed microdata, since what is reconstructed is the microdata *after* the noise has been added.

Swapping, the disclosure-avoidance approach used in the 2010 census, is a kind of input noise injection. In swapping, some of the attributes are exchanged, or *swapped*, between records. The advantage of swapping is that it has no impact on some kinds of statistics: if people are swapped only within a county, then any tabulation at the county level will be unaffected by swapping. The disadvantage of swapping is that it can have significant impact on statistics at lower levels of geography, and values that are not swapped are unprotected.

Option 3: Apply noise to the published statistics

This approach is called *output noise injection*. Whereas input noise injection applies noise to the microdata directly, output noise injection applies output to the statistical publications. Output noise injection complicates database

reconstruction by eliminating naïve approaches based on the straightforward application of SAT solvers. Also, even if a set of microdata is constructed that is mostly consistent with the published statistics, this microdata will be somewhat different from the original microdata that was collected. The more noise that was added to the tabulation, the more the microdata will be different.

When noise is added to either the input data (option 2) or the tabulation results (option 3), with all records having equal probability of being altered, it is possible to mathematically describe the resulting privacy protection. This is the basis of differential privacy.

Implications for the 2020 Census

The Census Bureau has announced that it is adopting a noise-injection mechanism based on differential privacy to provide privacy protection for the underlying microdata collected as part of the 2020 census. Following is the motivation for that decision.

The protection mechanism developed for the 2010 census was based on a swapping.[15] The swapping technique was not designed to protect the underlying data against a DRA. Indeed, it is the Census Bureau's policy that both the swapped and the unswapped microdata are considered confidential.

The 2010 census found a total population of 308,745,538. These people occupied 10,620,683 habitable blocks. Each person was located in a residential housing unit or institutional housing arrangement (what the Census Bureau calls "group quarters"). For each person, the Census Bureau tabulated the person's location, as well as

sex, age, race, and ethnicity, and the person's relationship to the head of the household—that is, six attributes per person, for a total of approximately 1.5 billion attributes. Using this data, the Census Bureau published approximately 7.7 billion linearly independent statistics, including 2.7 billion in the PL94-171 redistricting file, 2.8 billion in the balance of summary file 1, 2 billion in summary file 2, and 31 million records in a public-use microdata sample. This results in approximately 25 statistics per person. Given these numbers and the example in this article, it is clear that there is a theoretical possibility that the national-level census could be reconstructed, although tools such as Sugar and PicoSAT are probably not powerful enough to do so.

To protect the privacy of census respondents, the Census Bureau is developing a privacy-protection system based on differential privacy. This system will ensure that every statistic and the corresponding microdata receive some amount of privacy protection, while providing that the resulting statistics are sufficiently accurate for their intended purpose.

This article has explained the motivation for the decision to use differential privacy. Without a privacy-protection system based on noise injection, it would be possible to reconstruct accurate microdata using only the published statistics. By using differential privacy, we can add the minimum amount of noise necessary to achieve the Census Bureau's privacy requirements. A future article will explain how that system works.

RELATED WORK

In 2003 Irit Dinur and Kobbi Nissim[4] showed that the amount of noise that needs to be added to a database to prevent a reconstruction of the underlying data is on the order of $\Omega(\sqrt{n})$ where n is the number of bits in the database. In practice, many statistical agencies do not add this much noise when they release statistical tables. (In our example, each record contains 11 bits of data, so the confidential database has 77 bits of information. Each statistic in Table 3 can be modeled as a four-bit of count, a seven-bit of median, and a seven-bit of mean, for a total of 18 bits; Table 3 releases 126 bits of information.) Dinur and Nissim's primary finding is that many statistical agencies leave themselves open to the risk of database reconstruction. This article demonstrates one way to conduct that attack.

Statistical tables create the possibility of

SAT and SAT Solvers

The Boolean SAT problem was the first to be proven NP-complete.[9] This problem asks, for a given Boolean formula, whether replacing each variable with either true or false can make the formula evaluate to true. Modern SAT solvers work well and reasonably quickly in a variety of SAT problem instances and up to reasonably large instance sizes.

Many modern SAT solvers use a heuristic technique called CDCL (conflict-driven clause learning).[10] Briefly, a CDCL algorithm:

1. Assigns a value to a variable arbitrarily.
2. Uses this assignment to determine values for the other variables in the formula (a process known as unit propagation).
3. If a conflict is found, backtracks to the clause that made the conflict occur and undoes variable assignments made after that point.
4. Adds the negation of the conflict-causing clause as a new clause to the master formula and resumes from step 1.

This process is fast at solving SAT problems because adding conflicts as new clauses has

the potential to avoid wasteful "repeated backtracks." Additionally, CDCL and its predecessor algorithm, DPLL (Davis–Putnam–Logemann–Loveland), are both provably complete algorithms: they will always return either a solution or "Unsatisfiable" if given enough time and memory. Another advantage is that CDCL solvers reuse past work when producing the universe of all possible solutions.

A wide variety of SAT solvers are available to the public for minimal or no cost. Although a SAT solver requires the user to translate the problem into Boolean formulae before use, programs such as Naoyuki Tamura's Sugar facilitate this process by translating user-input mathematical and English constraints into Boolean formulae automatically.

database reconstruction because they form a set of constraints for which there is ultimately only one exact solution when the published table is correctly tabulated from a real confidential database. Restricting the number or specific types of queries—for example, by suppressing results from a small number of respondents—is often insufficient to prevent access to indirectly identifying information, because the system's refusal to answer a "dangerous" query itself provides the attacker with information.

CONCLUSION

With the dramatic improvement in both computer speeds and the efficiency of SAT and other NP-hard solvers in the last decade, DRAs on statistical databases are no longer just a theoretical danger. The vast quantity of data products published by statistical agencies each year may give a determined attacker more than enough information to reconstruct some or all of a target database and breach the privacy of millions of people. Traditional disclosure-avoidance techniques are not designed to protect against this kind of attack.

Sugar Input

Sugar input is given in a standard CSP (constraint satisfaction problem) file format. A constraint must be given on a single line of the file, but here we separate most constraints into multiple lines for readability. Constraint equations are separated by comments describing the statistics they encode.

Input for the model in this article is available online at https://queue.acm.org/appendices/Garfinkel_SugarInput.txt.

Faced with the threat of database reconstruction, statistical agencies have two choices: they can either publish dramatically less information or use some kind of noise injection. Agencies can use differential privacy to determine the minimum amount of noise necessary to add, and the most efficient way to add that noise, in order to achieve their privacy protection goals.

Acknowledgments

Robert Ashmead, Chris Clifton, Kobbi Nissim, and Philip Leclerc provided extraordinarily useful comments on this article. Naoyuki Tamura provided invaluable help regarding the use of Sugar.

References

1. Adam, N.R., Worthmann, J.C. 1989. Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys* 21(4), 515–556); http://doi.acm.org/10.1145/76894.76895.
2. Biere, A. 2008. PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling and Computation* 4, 75-97; https://pdfs.semanticscholar.org/7ea4cdd0003234f9e98ff5a080d9191c398e26c2.pdf.
3. Blum, A., Dwork, C., McSherry, F., Nissim, K. 2005.

Practical privacy: the SuLQ framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 128–138; https://dl.acm.org/citation.cfm?id=1065184.

4. Dinur, I., Nissim, K. 2003. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Principles of Database Systems*, 202–210; https://dl.acm.org/citation.cfm?id=773173.
5. Dwork, C., McSherry, F., Nissim, K., Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Conference on Theory of Cryptography*, 265–284. Berlin, Heidelberg: Springer-Verlag; http://dx.doi.org/10.1007/11681878_14.
6. Dwork, C., Nissim, K. 2004. Privacy-preserving datamining on vertically partitioned databases. In *Proceedings of the 24th International Cryptology Conference* 3152, 528–544. Santa Barbara, California: Springer Verlag; https://www.microsoft.com/en-us/research/publication/privacy-preserving-datamining-on-vertically-partitioned-databases/.
7. Heule, M.J.H., Kullmann, O. 2017. The science of brute force. *Communications of the ACM* 60(8), 70–79; http://doi.acm.org/10.1145/3107239.
8. Kleinberg, J., Papadimitriou, C., Raghavan, P. 2000. Auditing Boolean attributes. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 86–91; http://doi.acm.org/10.1145/335168.335210.
9. Kong, S., Malec, D. 2007. Cook-Levin theorem. Lecture, University of Wisconsin.

10. Marques-Silva, J., Lynce, I., Malik, S. 2009. Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability*, 131–153. Amsterdam, The Netherlands: IOS Press.
11. McCarthy, J. 1960. Recursive functions of symbolic expressions and their computation by machine, part I. *Communications of the ACM* 3(4), 184–195; https://dl.acm.org/citation.cfm?id=367199.
12. Metin, H., Baarir, S., Colange, M., Kordon, F. 2018. CDCLSym: Introducing effective symmetry breaking in SAT solving. In *Tools and Algorithms for the Construction and Analysis of Systems*, ed. D. Beyer and M. Huisman, 99–114. Springer International Publishing; https://link.springer.com/chapter/10.1007/978-3-319-89960-2_6.
13. Tamura, N., Taga, A., Kitagawa, S., Banbara, M. 2009. Compiling finite linear CSP into SAT. *Contraints* 14(2), 254–272; https://dl.acm.org/citation.cfm?id=1527316; http://bach.istc.kobe-u.ac.jp/sugar/.
14. Whitney, C.R. 1997. Jeanne Calment, world's elder, dies at 122. *New York Times* (August 5); https://nyti.ms/2kM4oFb.
15. Zayatz, L., Lucero, J., Massell, P., Ramanayake, A. 2009. Disclosure avoidance for Census 2010 and American Community Survey five-year tabular data products. Statistical Research Division, U.S. Census Bureau; https://www.census.gov/srd/CDAR/rrs2009-10_ACS_5yr.pdf.

Simson L. Garfinkel *is the Senior Computer Scientist for Confidentiality and Data Access at the U.S. Census Bureau*

and the Chair of the Census Bureau's Disclosure Review Board.

Related articles

Go Static or Go Home
In the end, dynamic systems are simply less secure.
Paul Vixie
https://queue.acm.org/detail.cfm?ref=rss&id=2721993

Privacy, Anonymity, and Big Data in the Social Sciences
Quality social science research and the privacy of human subjects requires trust.
Jon P. Daries, et al.
https://queue.acm.org/detail.cfm?id=2661641

Research for Practice: Private Online Communication; Highlights in Systems Verification
Expert-curated Guides to the Best of CS Research
Albert Kwon, James Wilcox
https://queue.acm.org/detail.cfm?id=3149411

John M. Abowd *is the Chief Scientist and Associate Director for Research and Methodology at the U.S. Census Bureau, where he serves on leave from his position as the Edmund Ezra Day Professor of Economics, professor of information science, and member of the Department of Statistical Sciences at Cornell University.*

Christian Martindale *is a senior at Duke University. He intends to pursue a career in management consulting or law.*