Data Obfuscation: Anonymity and Desensitization of Usable Data Sets

In some domains, the need for data privacy and data sharing conflict. Data obfuscation addresses this dilemma by extending several existing technologies and defining obfuscation properties that quantify the technologies' usefulness and privacy preservation.

David E. Bakken
Washington State University

Rupa Parameswaran and Douglas M. Blough
Georgia Institute of Technology

Andy A. Franz and Ty J. Palmer
Washington State University

Many application domains benefit from data sharing, which can range from passing data to a single party for specific calculations to aggregating data from many entities for data mining purposes. In some cases, however, data sharing is impractical due to privacy concerns. In the case of medical data, for example, even if the patient's name is stripped from the record, attackers can identify the patient by finding just a few accurate readings through other means. This is unacceptable for both legal and ethical reasons.

As the "Personalization and encryption methods" sidebar (p. 40) describes, many techniques have evolved for providing data anonymity and personalization. Anonymity management in personalization is a Web-based privacy service that operates on usage rather than data. Encryption techniques require that data be processed both before and after dissemination, and are incapable of offering different protection levels for different end users. Encryption techniques such as privacy homomorphisms are also susceptible to chosen-text attacks, in which a clever choice of plain text or cipher text reveals the key.

We are researching a family of mechanisms that addresses these limitations. Our data obfuscation techniques work within a data set, lowering individual data item accuracy in a systematic, controlled, and statistically rigorous way. Data obfuscation thus lets users disseminate sensitive data in a degraded form that, for many applications, permits sufficient calculation accuracy, but hides the data's most sensitive aspect. The main advantage of this technique over data encryption is its ability to provide multiple levels of data protection by distributing data that has been obfuscated by different amounts based on the end users' needs. In addition to techniques for desensitizing data, data obfuscation encompasses techniques such as data randomization,[1] data swapping,[2] and data anonymization.[3] To study and categorize the security provided by the various data obfuscation techniques, we introduce a reversibility property, which quantifies the strength of an obfuscation technique based on how difficult it is to reverse-engineer the obfuscated data set. We also discuss different obfuscation mechanisms and provide early performance results.

Existing data security approaches

The database community has long known that attackers can use characteristic formulas called *trackers* to compromise even a small query set.[4] In general, trackers let attackers calculate database statistics without requiring any advance knowledge of the database contents, as long as the system's queries use an arbitrary characteristic formula to select record subsets.

Data randomization systematically thwarts trackers from reconstructing a database through repeated queries. Data mining researchers use randomization techniques to develop an accurate aggregated data model without the data record's precise information. Data randomization mainly operates on a subset of database tables, fields, and records to maintain the database's statistical properties. The end user can manipulate data randomization to obtain the original values, after perturbing them by adding either a random variable or data discretization.[1] While researchers have applied data randomization mainly to databases and data mining, the concept is quite similar to random data obfuscation techniques; data randomization techniques are extended naturally to produce useable data sets that can be directly disseminated to end users, which data obfuscation supports.

Data anonymization attempts to classify data into fixed or variable intervals. Each data entry is then replaced by its class interval, and judicious interval choices can ensure that the statistical information is maintained. Latanya Sweeney and her colleagues developed a privacy protection method that guarantees that each data item will relate to at least *k* other entries, even if the records are directly linked to external information.[3] This technique takes a generalization and suppression approach to obtaining the required anonymity level: generalization replaces a value with a less specific value, while suppression does not release a value at all.

Data swapping intelligently swaps entries within a single field in a records set so that the individual record entries are unmatched, but the statistics are maintained across the individual fields.[2] Users can perform swapping such that the swapped values are close to each other, thus approximating the information in the nonobfuscated data records. This technique thereby provides a type of data obfuscation.

The three approaches mentioned earlier fall under the category of privacy preserving data mining techniques.[1] Data obfuscation is a generalization of existing techniques and has applications beyond data mining. The "Data obfuscation examples" sidebar (p. 41) lists example applications of data obfuscation. Data obfuscation provides a family of mechanisms that includes the three techniques mentioned above and many others, and also provides a standard for categorizing the various obfuscation techniques based on the properties and metrics we now describe.

Data obfuscation

Data obfuscation techniques feature three main properties: reversibility, specification, and shift. Figure 1 offers a high-level description of data obfuscation techniques and the various properties each supports. We now describe these properties, emphasizing reversibility, which is essential to data security.

Reversibility

Data obfuscation maps from the original datum, D, to a new one, D'. Reversibility describes how complex it is to reverse-engineer a given obfuscated data set, specifying the obfuscation technique's robustness in terms of data hiding. An obfuscation technique that is easy to reverse-engineer provides little data protection. On the other hand, an irreversible obfuscation technique requires that data be stored in its original form if that form needs to be preserved.

Reversibility techniques fall into four general categories: partial knowledge reversible, process reversible, combination reversible, and irreversible.

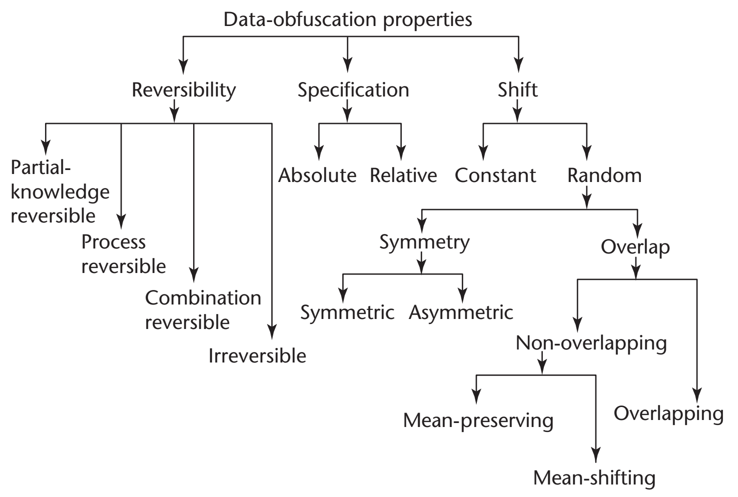

Figure 1. Data obfuscation properties. The three main properties—reversibility, specification, and shift—are each further defined by various subproperties.

Partial knowledge reversible. An obfuscation technique is partial knowledge reversible if the attacker can reverse-engineer the entire data set using a minimum number of original data set entries. The number of entries required depends on the obfuscation technique.

For example, matching the obfuscated data set with the original data set's known entries reveals a pattern if the obfuscation technique involves a constant offset to the original data set. Examples of such a technique are

$$y_i = x_i + \text{constant, and}$$
$$y_i = x_i * (1 + \text{constant}),$$

where x_i and y_i denote entry i in the original and obfuscated data set, respectively.

Process reversible. Knowing the obfuscation technique or one of its standard processes can lead to complete or partial reversibility of the obfuscated data set. Obfuscation techniques with this property are generally stronger than partial knowledge reversible techniques, but still vulnerable to reverse engineering. An example of a process reversible technique is

$$y_i = fn(x_i),$$

where x_i and y_i denote entry i in the original and obfuscated data set, respectively, and the inverse of function fn exists and is computable. In this situation, knowing fn is sufficient to reverse-engineer the obfuscated data.

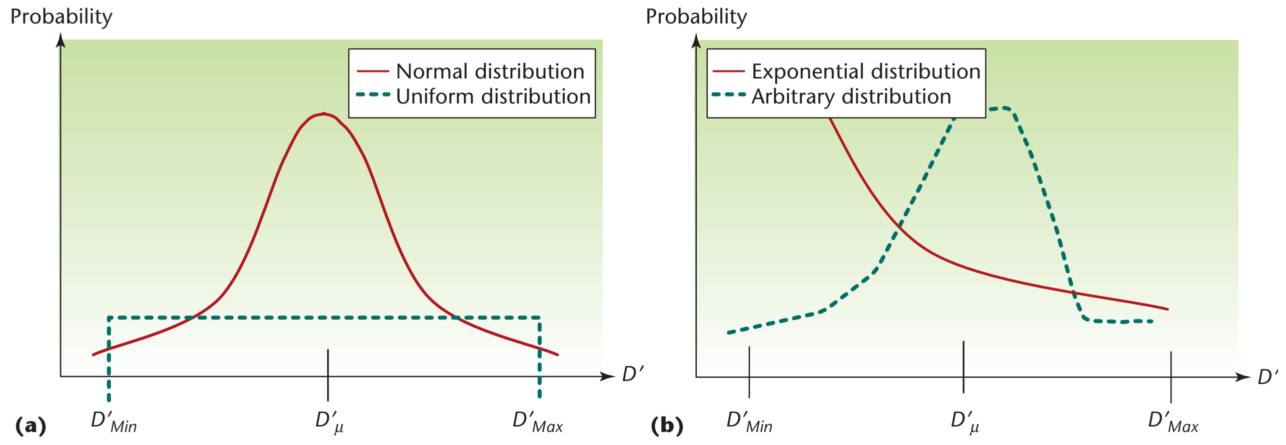

Figure 2. Symmetry in random obfuscations. (a) Symmetric curves with normal distribution (solid line) and uniform distribution (dashed line). (b) Asymmetric curves with exponential distribution (solid line) and arbitrary distribution (dashed line).

Most obfuscation techniques invoke pseudorandom number generators (PRNG) to generate random sequences. Knowing the sequence generation's random number seed and the PRNG facilitates the data set's reverse engineering. This special category of process reversibility is called *random number reversibility*. This property indicates that the original data set can be reverse-engineered if attackers know the PRNG and the obfuscation process. Once they obtain the sequence of random numbers, they can easily reconstruct the original data set by performing simple mathematical operations on the obfuscated data. The key item in preserving the data set's privacy lies in the PRNG's complexity. It is therefore essential that users avoid PRNGs that are easy to crack, protect the seed used to obfuscate a data set, and strictly avoid common seed values (such as 0 and 1). Examples of process reversible data obfuscation techniques are

$y_i = x_i + rv$, and
$y_i = x_i * rv$,

where x_i and y_i denote entry i in the original and obfuscated data set, respectively, and rv is a random variable. In this case, if an attacker obtains the random number sequence, the entire original data set is compromised.

Combination reversible. Obfuscation techniques with this property can be reverse-engineered if attackers have partial knowledge of the data obfuscation technique and the original data set. In this case, even if an attacker exploits the PRNG's weakness, for example, they'd still need to know some original data characteristic to obtain the original data set. An example of such an obfuscation technique is

$y_i = x_i*(1 + rv) + \text{constant}$,

where x_i and y_i denote entry i in the original and obfuscated data set, respectively, and rv is a random variable. In this case, attackers could combine knowledge of the random obfuscation process (including the PRNG and seed) with some original data values to reverse-engineer the constant and obtain the entire original data set.

Irreversible. Techniques in this category are impossible to reverse-engineer. Data obfuscated with such techniques can't be mapped back to their corresponding original values. Deterministic data obfuscation techniques that result in a many-to-one mapping of original data entries to obfuscated data entries are irreversible. For example,

$$y_j = y_{j+1} = k = y_{ik-1} = y_k = \sum_{j}^{k} x_i \,/\, (k - j + 1),$$

where x is the original data sequence and $[x_j, \ldots, x_k]$ is a subsequence of x. Here, the output for all subsequence values is the subsequence mean. With irreversible techniques, there is an inherent loss of information. The natural analogy here is to lossy compression techniques, which make it impossible to exactly recover the original data. The obvious disadvantage is that the original data set must be stored, as it cannot be regenerated even by authorized users.

Other properties

The specification and shift properties define specific properties within an obfuscation mechanism. With data anonymization, specification and shift refer to the interval's size; with data swapping, they refer to the distance between the nearest neighbors for swap selection. Specifically, the specification property defines the obfuscation parameter and the shift parameter defines the obfuscation

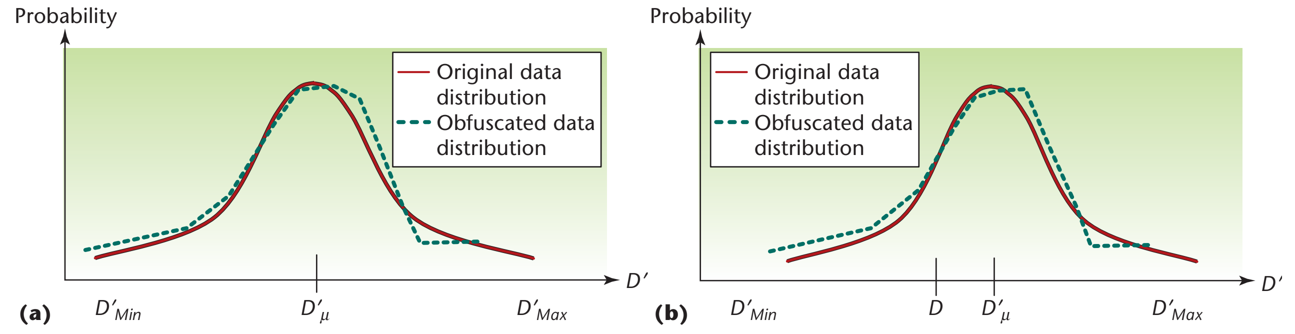

Figure 3. Mean preservation in overlapping random obfuscations. (a) A mean-preserving obfuscation ($D'_{\mu} = D$). (b) A mean-shifting obfuscation (where $D'_{\mu} \neq D$).

process. The specification property is further categorized as either absolute or relative:

- *Absolute obfuscation* implies that the magnitude of change to be introduced to the datum is independent of the datum's magnitude.
- *Relative obfuscation* gives a percentage by which the data should be changed.

The shift property describes the nature of the obfuscation process. Data can be obfuscated by a constant or random amount:

- A *constant obfuscation* changes each datum in a data set by the same specified amount, whether absolute or relative.
- A *random obfuscation* will change different data items by different amounts specified in statistical terms.

Random obfuscation's properties all involve reasoning about the probability density function (pdf) of a given obfuscation's random distribution. Although we can use a wide range of pdf functions with random data obfuscation, our discussion applies to the overall category. In all cases, the pdf is truncated at $D' \in [D'_{min}, D'_{max}]$, because most applications require a bounding on the largest possible data perturbation. In implementation terms, if a pdf generates a value outside this range, the system calls its random number function repeatedly until the value fits within the range.

Random obfuscation has two sub-properties: *symmetry* and *overlap*. The obfuscation is symmetric if it uses a symmetric pdf, and is asymmetric otherwise. Figure 2a shows symmetric curves, one using a normal distribution and another using a uniform distribution. Figure 2b shows asymmetric curves, one using an exponential distribution and the other using an arbitrary one.

Overlap indicates the range $[D'_{min}, D'_{max}]$ that D' can be mapped onto, as Figure 3 shows (the figure doesn't show pdfs as they are not germane to overlap definitions). A nonoverlapping random obfuscation is such that $D \notin [D'_{min}, D'_{max}]$, that is, D' cannot equal D. Consider, for example, the transformation that truncates such that $D' \in [D'_{min}, D)$ or $D' \in (D, D'_{max}]$, which is a minimum perturbation nonoverlapping obfuscation.

Overlapping obfuscations differ in their mean preservation—that is, whether the arithmetic mean of the truncated distribution, D'_{μ}, equals D. As Figure 3a shows, a mean-preserving obfuscation will have $D'_{\mu} = D$. Figure 3b shows a mean-shifting obfuscation, in which the truncated distribution doesn't equal D.

This set of data obfuscation properties is rich enough for a wide range of application programs, and all of them can be implemented. Any function can be devised with a random obfuscation, but the resulting properties are unknown in most cases. Also, we could derive further properties for a given random distribution, but that is beyond the scope of this article.

Example primitives

As Table 1 shows, we use various symbols to specify several primitives that can implement the obfuscation properties.

Constant primitives. A constant obfuscation primitive family is fairly simple:

$$D' = \textbf{Constant\_ob}\ (D, exact\_shift) = D + exact\_shift,$$

in which *exact_shift* is the amount to obfuscate by, which can be relative (a percent) or absolute (a fixed amount). By definition, constant obfuscation shifts all data set items by this same relative or absolute specification; there are no random variables.

Random primitives. Another simple primitive, the random obfuscation family makes various changes to data set items, with a known arithmetic mean and bounded change:

Table 1. Symbols for obfuscation primitives.

SYMBOL	DEFINITIONS
exact_Pct	Exact percent of change based on original *D*. This primitive is used when max_Pct and min_Pct are not used; it has a range of 0 to 100.
minΔchng	*D**(min_Pct/100).
maxΔchng	*D**(max_Pct/100).
exactΔchng	*D**(exact_Pct/100).
$X(-1,1)$, $Y(0,1)$	Random variables that truncate their distribution with the specified ranges. *Z* can be either *X* or *Y*.
dist	A given distribution, such as normal.
dist_params	Any parameters that the given distribution requires to specify the random variable's shape.

D´ = ***Random_ob*** *(D, scalar, dist, dist_params) = D + (Y (dist (dist_params)) * scalar).*

Here, *scalar* could be constant, relative (maxΔchng or minΔchng), or based on min_shift/max_shift. It scales the distribution—which *Y* truncates to [0, 1]—to the appropriate range. We can construct random primitives to apply their statistical shifting across a whole data set or with a given statistical shift for each data item. The former approach shifts statistics across the entire data set, while the latter shifts multiple obfuscated copies of the same data item with the given statistical parameterization.

Combination primitives. Generally, we can compose all primitives independently and know, a priori, the resulting D''_μ. However, it can be useful to explicitly combine primitives. While composing the above primitives can desensitize them, composition also allows the direct specification of the total obfuscation's parameterization. If we compose multiple primitives, for example, we can derive D^*_μ given the obfuscation algorithms and their parameters. However, even for a given list of obfuscation algorithms, deriving how they should be parameterized given the desired resulting D^*_μ can be computationally intractable or even impossible, depending on the set of primitives.

Obfuscating data structures

So far, we've mainly discussed obfuscation on a single datum, or at most on a data array in which each item might be randomly obfuscated to different degrees (with well-chosen bounds and statistical properties) to make reverse engineering more difficult. However, these obfuscation approaches also directly extend to many different data structures. Such an extension offers great opportunities to take advantage of each data structure's semantics.

For example, in a simple data structure with fields for coordinates in two or three dimensions, multiple fields constitute a coordinate. We can take advantage of this by obfuscating the fields in the same way, or in different ways that take advantage of the application's semantics. In the latter case, the importance of one dimension, such as altitude, might vary, and thus we might obfuscate it more or less than the other two dimensions. Alternately, data semantics might require that we obfuscate multiple data items while considering others in the set. Take, for example, RGB, which specifies a color by its components. In some cases, we'll want to change all three at once, because a user's perception of the overall change must account for the current values of all three and change them appropriately. In other cases, we'll want to perform a nonlinear transformation on an RGB (or digitized audio) structure, because the visually perceived change is not linear with the given value's change, and a data structure containing an image can obfuscate pixels based on adjacent pixels' values. It's also possible to obfuscate using an application's general semantic information, and perhaps even use metadata (from the data structure or a remote database) to identify the object that a group of adjacent pixels represents to suggest obfuscation approaches. Such an approach minimizes data perturbation, while being difficult or virtually impossible to reverse-engineer.

Another example comes from the network security realm, where it can be useful to share computer attack and intrusion data both for real-time analysis and post facto forensics. However, many companies are hesitant to share such data because it identifies them (via IP addresses, for example) as having suffered computer security failures, which can be bad for business. The alert logs produced by intrusion detection systems and other alert mechanisms contain sensitive information about the compromised system. Such information includes the source and destination IP addresses and the system configuration, which are likely targets for privacy invasion.[5] To address this, we can obfuscate IP addresses that are stored as simple unsigned integers. Depending on the obfuscation approach, we can preserve both the company's anonymity and the properties that make the data useful for data mining and attack and forensic analyses.

Table 2. Random data obfuscation results.

TEST MACHINE PRIMITIVES	PIII 450MHZ 256 RAM WINDOWS 2000	AMD 1.4GHZ 256 DDR RAM WINDOWS XP PRO	AMD 1.8GHZ 512 DDR RAM WINDOWS XP PRO
Constant_ob() (absolute)	0.39 μs	0.10 μs	0.06 μs
Constant_ob() (relative)	0.47 μs	0.12 μs	0.08 μs
Random_ob() (absolute)	12.17 μs	3.03 μs	2.36 μs
Random_ob() (relative)	12.24 μs	3.04 μs	2.37 μs

Obfuscating IP address properties is an open research area, but we can offer a few initial examples. In most situations, the obfuscation must preserve IP-specific information embedded or inherent in the integer—such as its class (A, B, or C) and subnet size—while transforming higher bits to blur the company's identity. Such transformations might need to preserve other properties, such as the number of hops from the company to an Internet backbone provider and the company's different operating systems.

Many scientific applications can use data structure obfuscations, representing large arrays using sparse representations that store the indices along with the data. Obfuscating the indices can provide much greater security, but for some applications, the data set becomes unusable. Also, in general, indices must be obfuscated differently than the datum they reference. Databases often store records using trees, and in many cases we can change the tree's shape (a kind of non-numerical obfuscation) while preserving the data set's usability.

Implementation

There are many ways to implement data obfuscation. For floating-point numbers, bit-level manipulations can be performed on the mantissa to provide relative changes. This provides less granularity than arithmetic computations, but it can be implemented in hardware with a much smaller footprint than a general floating-point arithmetic module, and it's also potentially faster.

In many situations, obfuscating every datum in a data set is undesirable and wastes resources. It can also degrade security. Consider the situation in which a data item is obfuscated, but an adversary can accurately derive its value. An example of this is the precise position of a fixed and publicly visible structure, such as a water tower. In this case, an adversary can verify the position with great accuracy. Attackers can use this position data to determine how much the datum has been obfuscated. By collecting such information about multiple obfuscated items, adversaries might ascertain the obfuscation algorithm and its parameterization. They could thus de-obfuscate the other data items in the data set that could not be independently verified.

Experimental evaluation

We now provide an experimental evaluation of some data obfuscation mechanisms.

Performance metrics

To measure an obfuscation technique, we can use metrics for reversibility, usability, perturbation, and performance. *Reversibility* metrics let us quantify the reverse-engineering capability of obfuscated data. We define four metrics for reversibility, based on different aspects of the data retrieval:

- *Targeted individual data retrieval* denotes the minimum amount of information necessary and sufficient to reverse any individual entry in a given obfuscated data set.
- *Subset reversal* defines the minimum amount of information necessary to reverse-engineer a subset of the obfuscated data set.
- *Complete reversal* denotes the information needed to completely reverse obfuscated data.
- The *time complexity* of reverse engineering a data set determines an encryption mechanism's strength; we can use it to measure reversibility as well.

Usability metrics describe how usable the obfuscated data set is. Such metrics limit how much a data item can be obfuscated while retaining certain statistics on the set and its subsets. *Perturbation metrics* measure how much the data actually shifts compared to the target shift amount. Application-level perturbation measures the perturbation's impact on the application using the perturbed data. Finally, *performance metrics* measure how a specific obfuscation performs in terms of classical measures such as execution time, throughput, and so on.

Experiments

We evaluated the execution time of several primitives, running the results on various desktop computers using Windows 2000 or Windows XP. We implemented two different tests with two different obfuscation methods.

In the first test, we obfuscated an array of randomly generated 32-bit floating-point numbers using a straight mathematical implementation of the primitives. We performed the random data obfuscation tests using a normal distribu-

Personalization and encryption methods

Managing anonymity while sharing knowledge to servers (Masks), is an integrated Web-based tool that addresses Web users' privacy concerns while also considering their desire for personalized Web services.[1] Masks addresses privacy at the data-collection level, working as a proxy that provides users with a temporary identification for interacting with a Web site The Lucent personalized Web assistant (www.bell-labs.com/projects/lpwa) lets users create and use unique identities to register with Web sites that provide personalized services, while keeping their true identity private. Onion (www.onion-router.net) is a Web-based system that resists packet analysis and snooping by removing identification information from a data stream's routing information. These services provide anonymity of Web usage, but don't operate on data.

Data encryption complements data obfuscation in that it secures the data only against unauthorized users, offering authenticated users the original data. However, data encryption doesn't provide different data protection levels to different user categories. Privacy homomorphisms are a subcategory of encryption algorithms that allow additive and multiplicative computations on encrypted data.[2] Researchers have devised several PH techniques to provide different computation capabilities. However, all existing PH encryptions have been broken by chosen-plain-text attacks and cipher-text attacks[3] (an encryption is insecure to chosen-plain-text attacks when a clever choice of the plain text to be encrypted reveals key information; in a cipher-text attack, selectively choosing cipher-text to be decrypted can reveal the decryption key). In such cases, an end user with access to a part of the original data set can crack the key and retrieve all of the sensitive data.

Finally, obtaining encrypted data statistics requires a third party to perform the statistical computations, which leads to the need for more trusted resources.[4] For trend analysis and statistical and inference-based computations from data sets, encryption-based security schemes add complexity, yet don't guarantee absolute data protection.

References

1. L. Ishitani, V. Almeida, and W. Meira, "Masks: Bringing Anonymity and Personalization Together," *IEEE Security & Privacy*, vol. 1, no. 3, May/June 2003, pp. 18–23.
2. R. Rivest, L. Adleman, and M. Dertouzos, "On Data Banks and Privacy Homomorphisms," *Foundations of Secure Computations*, eds. R.A. DeMillo et al. Academic Press, 1978, pp.169–179.
3. F. Bao, "Cryptanalysis of a Provable Secure Additive and Multiplicative Privacy Homomorphism," *Int'l Workshop Coding and Cryptography*, 2003; www.i2r.a-star.edu.sg/icsd/publications/BaoFeng_2003_WCC.pdf.
4. J. Domingo-Ferrer, "Privacy Homomorphisms for Subcontracting Statistical Computation," *Proc. Third Int'l Seminar Statistical Confidentiality*, Eurostat-Statistical Office of the Republic of Slovenia, 1996, pp. 184–191.

tion and parameters $X(-1,1)\,Y(0,1)$. Table 2 shows the results. Clearly, the basic primitives' execution time is insignificant: We can obfuscate a one-million-element data set in a few seconds with state-of-the-art processors. We're currently evaluating more complex obfuscation mechanisms.

In our second test, we didn't implement the primitives using an arithmetic formula; we produced the desired obfuscation by directly manipulating specific bits in the floating-point number's mantissa. As expected, the mathematical implementation was significantly faster than the bit-level manipulation (we omit the actual numbers here for brevity's sake). This is due to overhead incurred by the table lookup that we used to determine which mantissa bits to manipulate to induce the desired change. We haven't yet optimized the algorithm for this table lookup; once we do, we expect execution times to improve. We also expect that this bit-manipulation approach to obfuscation will excel when implemented in hardware. Such an implementation should prove faster and, more importantly, it should be much smaller and potentially harder to reverse-engineer than the current implementations.

Almost all of our future research involves working toward a broad and fundamental understanding of the inherent trade-offs between security, usability for specific applications, reversibility, and implementability. A key research area will be to evaluate a sufficiently large set of random distributions to better understand their inherent reversibility as well as their usability in different application domains. Another key area is to try to provide statistical properties other than the arithmetic mean (such as variance, for example). These properties have the potential to greatly increase the obfuscated data's usability to some applications, but—except for well-chosen distributions and parameterizations thereof—they can rarely be implemented. □

Acknowledgments

The US Air Force Research Laboratory's Software Protection Initiative funds this research through contract F33615-02-C-1292. We thank Deb Frincke for suggesting intrusion data sharing as a data obfuscation application and Marty Stytz for suggesting simulation data sharing and for making numerous other suggestions on our research. We thank Alan Genz and Murali Medidi for their comments on data obfuscation properties.

References

1. R. Agrawal and S. Ramakrishnan, "Privacy-Preserving Data Mining," *Proc. ACM SIGMOD Conf. on Management of Data*, ACM Press, 2000, pp. 439–450.
2. S. Gomatam and A. Karr, "Distortion Measures for Cat-

Data obfuscation examples

Data obfuscation is applicable in many areas, including those involving highly sensitive information that must be disseminated. Examples here include medical records, electronic billing transactions, and military information.

Medical records

When dealing with a patient's medical records, data privacy is of the utmost concern. This requirement fundamentally conflicts with the scientific method's basic repeatability needs, which require that research results be independently validated and extended. Even in the most controlled circumstances, disseminating patient medical records greatly increases the odds of data misuse. In particular, someone who obtains a data set can identify a particular individual's medical record if he or she can independently acquire even a few fields with relatively high precision. Judicious medical record obfuscation can preserve data usability for medical research while preserving data anonymity, and thus thwart such attacks. It can work because many medical studies use large collections of confidential patient records, cross-correlating the arithmetic mean (and possibly other statistical properties as well) of different data items such as blood pressure, cholesterol, time spent exercising, and the amount of fat in a person's diet. In other cases, small statistical perturbations of each data item are within experimental tolerances.

Billing transactions

Obfuscation will also be useful in electronic billing transactions, such as electronic transponder-based toll collection for toll roads.[1] This business operates using tollbooth sensors to track users' individual transponders and bill them accordingly. The tollbooth operator wants to preserve transaction privacy so its customers don't feel like "Big Brother" or anyone else is tracking their movements. However, credit-card issuers might purchase tollbooth operator's information and cross-reference it with their billing statements to find tolls charged to a supposedly anonymous transponder company and thereby determine a particular customer's driving habits. Obfuscating the data contained in the billing statements before selling it to a credit-card issuer would prevent data from being cross-referenced to driving habits. It would thus protect the driver's privacy, while at the same time allowing the tollbooth operator to release the data, which would still be useful for many kinds of (non-invasive) aggregation and data mining.

Military information

Military weaponry is another highly sensitive area that could benefit from data obfuscation. Obfuscating statistical information about military-grade weapons' capability and technical parameters might make it possible for third parties not holding top security clearance—such as academics and consultants—to demonstrate a weapon's effectiveness and feasibility without releasing the data's full accuracy and thus its full military value. This technique would help prevent an adversary from using the distributed information to develop the weapons or countermeasures against them.

Reference

1. J. Warrior, E. McHenry, and K. McGee, "They Know Where You Are," *IEEE Spectrum*, vol. 40, no. 7, July 2003, pp. 20–25.

egorical Data Swapping," tech. report 131, US Nat'l Inst. Statistical Sciences, Jan. 2003.
3. L. Sweeney, "K-Anonymity: A Model for Protecting Privacy," *Int'l J. Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, 2002, pp. 557–570.
4. D. Denning and M. Schwartz, "The Tracker: A Threat to Statistical Database Security," *ACM Trans. Database Systems*, vol. 4, no. 1, March 1979, pp. 76–96.
5. P. Lincoln, P. Porras, and V. Shmatikov, "Privacy-Preserving Sharing and Correlation of Security Alerts," *Proc. 13th Usenix Security Symp.*, Usenix Assoc., 2004, pp. 239-254.

David E. Bakken *is an associate professor of computer science at Washington State University in Pullman, Washington. His research interests include distributed computing, middleware, fault tolerance, and security. He is the lead principal investigator on the GridStat project, which is developing next-generation communication infrastructures for the electric power grid and other critical infrastructures. He is a member of the IEEE and ACM. Contact him at bakken@wsu.edu.*

Rupa Parameswaran *is a PhD student at the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta, where she participates in a Raytheon-sponsored project to develop a fault-injection tool for middleware applications. Her research is focused on developing quantifiers for privacy-preserved sensitive data sets; her general interests include security, privacy, data mining, and middleware reliability. She received an MS in electrical and computer engineering from Georgia Institute of Technology. She is a student member of the IEEE and the ACM. Contact her at rupa@ece.gatech.edu.*

Douglas M. Blough *is a professor of electrical and computer engineering at the Georgia Institute of Technology. His research interests include distributed systems, dependability and security, and wireless ad hoc networks. He was program co-chair for the 2000 International Conference on Dependable Systems and Networks and the 1995 Pacific Rim International Symposium on Fault-Tolerant Systems. He was associate editor for* IEEE Transactions on Computers *from 1995 through 2000, and is currently associate editor for* IEEE Transactions on Parallel and Distributed Systems. *Contact him at doug.blough@ece.gatech.edu.*

Ty J. Palmer *is an independent consultant in Selah, Washington. His research interests include data obfuscation and middleware. Contact him at win270_64@hotmail.com.*

Andy A. Franz *is a system programmer in Selah, Washington. His research interests include data fusion and voting in middleware and data obfuscation. Contact him at druid_galiphile@yahoo.com.*