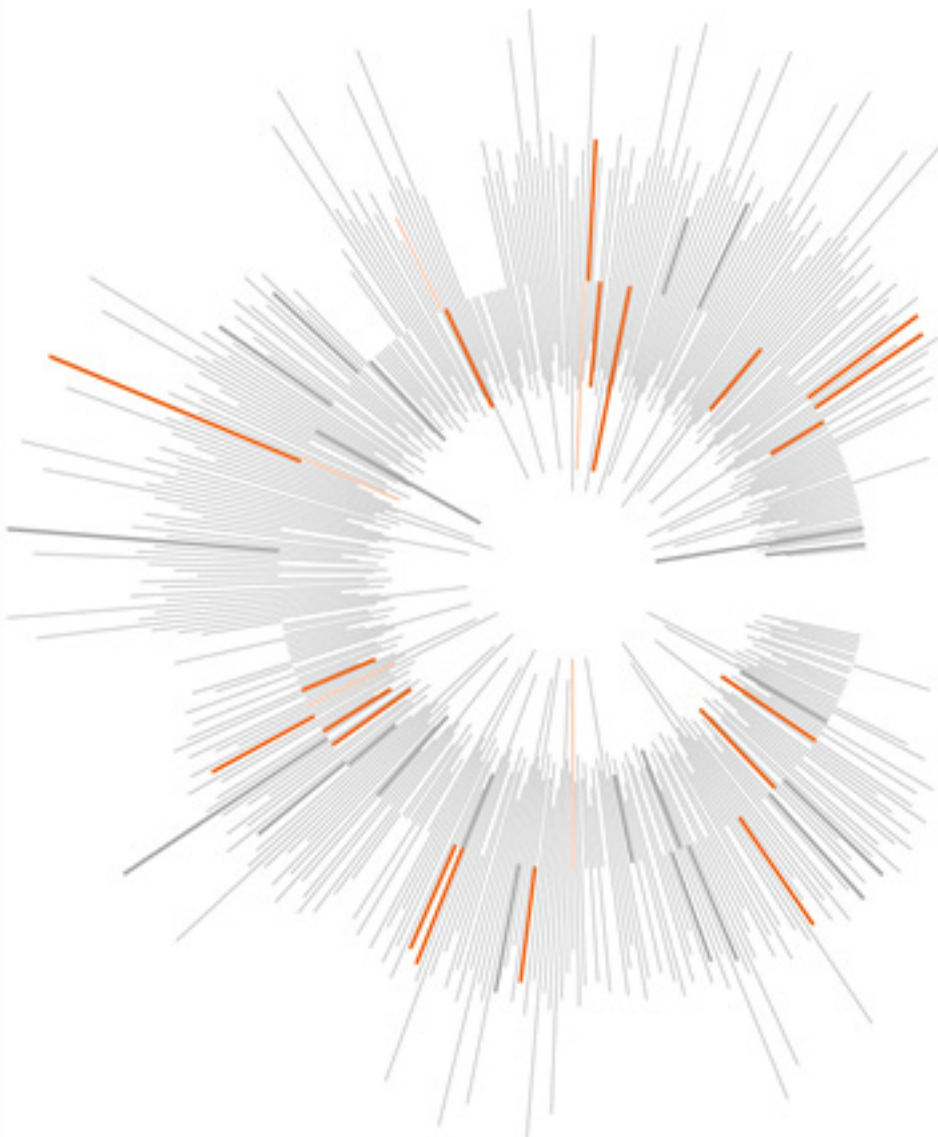


The Mathematics of Data Masking

This whitepaper outlines a simple scheme whereby the effectiveness of data masking is evaluated with respect to the effort required. Different 'orders' of masking are explored, and a mathematical basis for modelling the masking process is developed. Example models for different kinds of system are demonstrated.

By Llyr Jones and Richard Fine



Contents

Introduction.....	3
Orders of masking.....	3
Efficacy of mask operations with respect to orders.....	7
Applying the Crackability model.....	9
Conclusion.....	11
About Grid-Tools.....	11

Introduction

In this document we outline a simple scheme whereby the effectiveness of data masking is evaluated with respect to the effort required. The key quantity we will be concerned with is the ‘crackability’ of the data: we define this as the proportion of interesting questions that have identical answers when asked of each of the masked and the production datasets, usually expressed as a percentage. Note that this does not purely pertain to de-identification; we are interested in protecting all potentially sensitive or damaging data.

Orders of masking

It is evident that not all data masking operations have the same effectiveness, nor do all data masking operations seek to plug the same weaknesses in the data. For this reason, we will talk about the order of a data masking operation:

1. First order – this relates exclusively to the content of the data. Masking of fields such as names, addresses, social security numbers, dates of birth, telephone contact details, and so forth fall under this category. Usually we consider first order masks to be the utmost minimum that any masking solution should implement, and these typically require the least amount of investment.

Production Data				Test Data		
Peter	McNamara	£100	→	John	Smith	£94
Michael	Bowles	£152		Delia	Jones	£23
Julie	Andrews	£30	→	Paul	Brown	£181

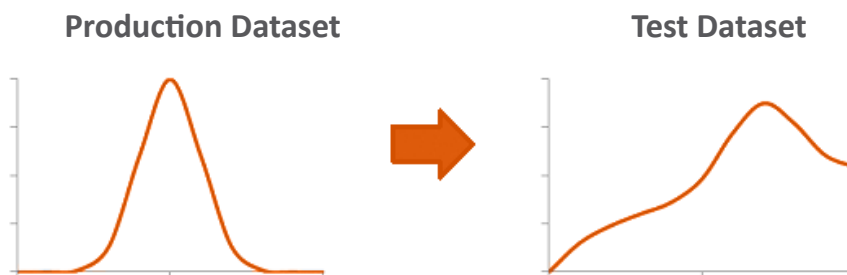
The need for these masks is usually the easiest to identify, as direct uses of sensitive data are usually well-known within the project. Projects that use more data of this type (e.g. HCM, CRM, healthcare) will take longer to mask than projects that do not (e.g. ERP, supply chain management, project management) as there is more work to be done, but in both cases progress happens at a fairly constant rate.

2. Second order – this involves a combination of the content and the structure of the data. These are fields that are sensitive in their own

right, but do not come under first order masking because they are fields which are used multiple times across the dataset (usually in different tables). Account numbers can be considered as a common example of this – they appear more than once (say, in ‘credit card’ and ‘orders’ tables) and are related via a foreign key constraint.

As with first order masks, the need for these masks is again fairly easy to identify, but masking them is more time-consuming as the changes are more widespread, making referential integrity harder to maintain. Systems that are more structurally complex take longer to mask as there are more changes to be made.

3. Third order – these are purely structural masks and relate to how the data is related both between tables and within tables. The primary metric here is the invariance of a query with respect to outliers, trends, aggregate queries, and so forth. Fields that usually fall under this category are currency amounts (be they order amounts, salaries etc.), product order quantities, and any other field that can give rise to any trend or can be subjected to an aggregate query. It is also important to note that these kinds of masks are hardly straightforward and do incur a steep computational penalty – but are required to maximise the security of the data and minimise liability.



This third order of masking is most commonly overlooked by users of data masking techniques, partly because it is difficult to identify which correlations can yield valid data, and partly because they fail to acknowledge that data masking is about more than regulatory compliance: many of the trends and statistics available through correlation are of serious commercial value, and could be very damaging if obtained by one’s competitors.

4. Fourth order – these masks are also purely structural, but rather than dealing with interrelationships between individual rows of data, they address information that can be obtained from the database schema itself. This entails using things like the data types of fields, foreign-key relationships between tables, or the very fact that a given field exists, to answer queries about the business’s operating conditions and expectations. For example, the fact that the dataset includes a table for recording which products a customer browsed before making a purchase demonstrates that the company is gathering such data, most likely because they have a strategy of using such data to improve their customer targeted marketing.

The need for these masks is often difficult to identify, as it is difficult to predict which aspects of the business’s operating conditions are sensitive. Most of the operating conditions for a business are common to any business within that field, or can be easily determined through other means, so masking would be pointless. Implementing masks at this level also has severe implications for the verisimilitude of testing; while the actual data content of the dataset will be highly variable and can thus be masked or synthesized without invalidating the coverage given by the tests, the schema usually does not vary at all, such that it is impossible to create a ‘test schema’ that masks sensitive information whilst still being a potential image of a production database.

5. Fifth order – these masks deal with meta-information, such as the RDBMS used to store data within the company, or the testing methodology being employed. For instance, knowing which RDBMS is used to store the test data is usually a good indicator of which RDBMS is used to store production data, as the tests would commonly seek to replicate as much of the RDBMS configuration as possible. This could be useful information to anyone seeking to illegally access (i.e. ‘hack’) the production data, by exploiting security flaws in the RDBMS.

In practice, masking at this level is never performed because the information is not sufficiently sensitive. The majority of companies have policies in place that mitigate the risk posed, such as a rigorous maintenance policy to keep the RDBMS updated with the latest security patches.

Note that the alternative approach to data masking – using synthesized data – circumvents the first three levels of masking. First- and second-order masking operations, which hide values that are directly sensitive, are unnecessary as there are no directly sensitive values in the database (except by sheer coincidence, and such values are both easy to remove, and extremely difficult for an attacker to single out amidst the fictional values). Third-order masking is also mostly unnecessary, as the aggregations over the data are extremely unlikely to yield the same results as in the production database. Fourth- and fifth-order masking is still an issue, but can be performed in exactly the same way as for data masking, by changing the schema or the database configuration.

Logically, it is also necessarily the case that the test data must have some properties in common with the production data set, particularly at the fourth and fifth orders: for example, it is necessary that the schema be broadly the same in both systems, otherwise the testing bears increasingly little relation to how the production system will operate. However, it is sometimes the case that properties must be preserved at lower levels of masking – most commonly, broad trends in the data, potentially hidden by third-order masking.

As these requirements are always present, it is impossible to render a data set completely uncrackable, regardless of whether masking or synthesis is used; as per our definition of ‘crackability,’ we are observing that we require that certain queries return identical results when carried out on masked and production data, and as such the proportion of such queries is necessarily nonzero.

As fourth- and fifth-order masking is usually not worth implementing, in that the information is usually not sufficiently sensitive for it to be worth the fictionalization of the test results, the rest of this paper will focus solely on the masking operations that alter the content of the dataset.

Efficacy of mask operations with respect to orders

A given masking technique will have different effects on ‘crackability’ according to which order the field falls under. Suppose we’re dealing with an ‘employees’ table, where an employee’s name and address are stored along with their monthly salary. Suppose the salary field is masked by shuffling them around – i.e. the actual numbers do not change, but each number assigned to a different person. If we wanted to know the salary of a single employee, then we could consider the mask to be somewhat effective – the efficacy of the mask with respect to the first order is remarkably good. However, now suppose we were interested in the total salary bill of the company – to find this out we would compute the sum of salaries over all employees. Here the shuffling operation is rendered useless: the summation function does not rely on the position of the numbers, just the actual numbers themselves. So we would say that the efficacy of the mask with respect to the third order is very poor (or useless).

This motivates the following: let o denote the order of the mask (1, 2, or 3) and let $e(i, o)$ denote the efficacy of mask i with respect to order o . We will restrict this quantity to the range $[0, 1]$, where 0 denotes ‘completely useless’ and 1 denotes ‘completely effective’. We then define the combined efficacy of mask i as the product of each $e(i, o)$ over each possible order, i.e.

$$e(i) = \prod_{o=1}^3 e(i, o)$$

And the aggregate efficacy E is defined as the average of the $e(i)$, i.e.

$$E = \frac{1}{n} \sum_{i=1}^n e(i)$$

(where n denotes the total number of masking operations.)

Up until now we have not attempted to quantify the effort required by each masking operation. We will assume this is a function of the order, as opposed to the masking function itself – the justification of this comes from the previous section on masking orders. We express the effort as a relative quantity: let $p(i, o)$ denote the effort (or power) required to carry out a mask i on an o -th order field, restricted to the range $(0, 1]$ where 0 denotes no effort and 1 denotes maximum effort. Since the notion of ‘maximum effort’ is arbitrary to an extent, using a relative quantity such as this allows us to compare each mask. Similar to before, we define the effort required for each order, $p(o)$, to be:

$$p(o) = \sum_{i=1}^n p(i, o)$$

Similar to before, we define the total effort P to be the sum over each order, i.e.

$$P = \sum_{o=1}^3 p(o)$$

Relating the efficacy to the effort – logistic function

Now we make a rather bold assumption: we propose that the ‘crackability’ C (which is defined as $C = 1 - E$) is a function of P (the effort required), and that it approximately follows a logistic curve, starting at $C(0) = 100$ and tends to 0 as P tends to infinity. In fact, we propose that the curve is a series of three approximating linear cords: the first cord (the steepest) denotes the effect of first order masking; the second cord (less steep) denotes the effect of second order masking; and the third cord (the least steep) denotes the effect of third order masking. The justification for this is laid out in the section on masking orders and will not be reiterated here.

The equation we propose to use is the following (a form of the logistic equation):

$$C(P) = \frac{100a}{100b + (a-100b)e^{-aP}}$$

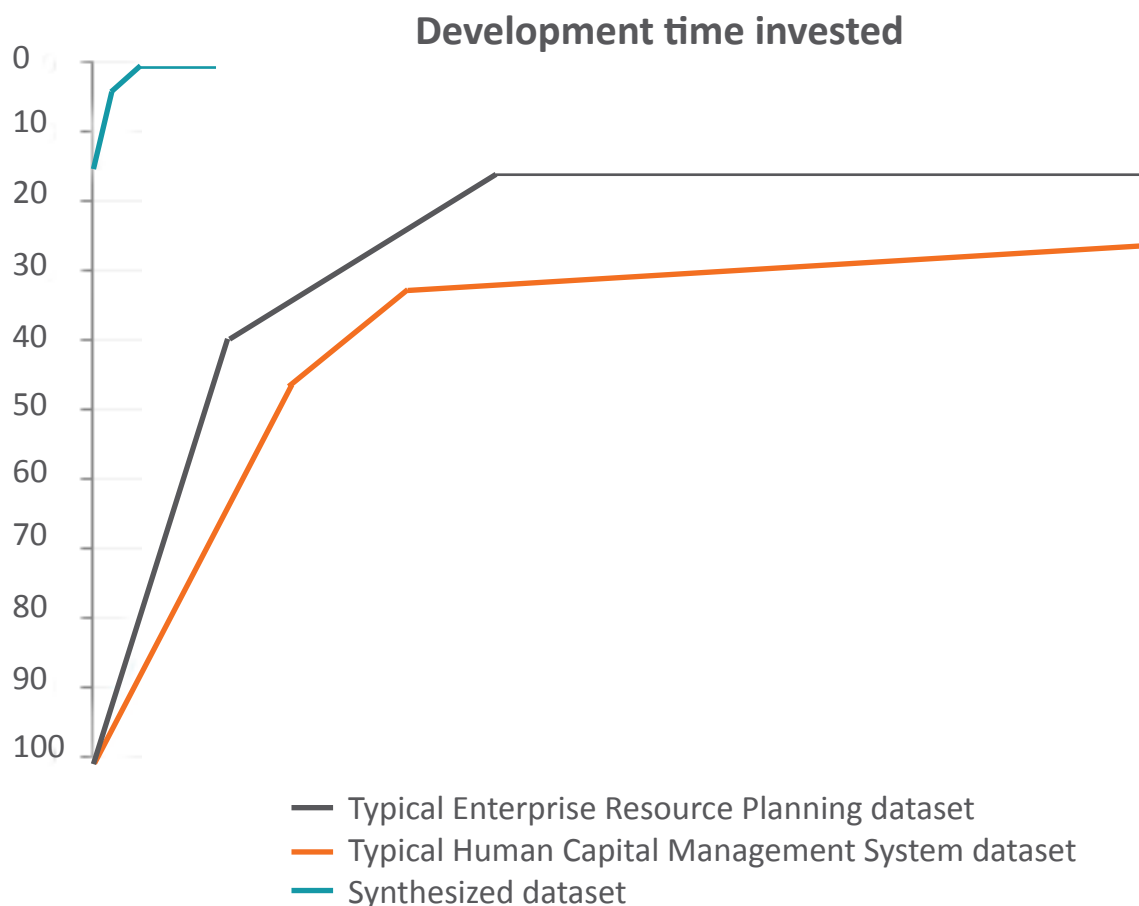
Here a , b are real-valued positive constants between 0 and 1, and e in this context denotes the exponent function. It is easy to show that, as P tends to infinity,

$$C(P) \rightarrow a/b$$

Thus a/b can be thought of as the theoretical minimum of the crackability. As discussed, the crackability can never reach zero – and it is clear that the equation reflects this.

Applying the Crackability model

To demonstrate how the crackability model can be used to understand the relationship between the efficacy of masking and the effort required to implement it, we have modelled the data masking process as applied to two common classes of production system. For comparison, we have also graphed the crackability of a synthetically generated dataset.



The typical ERP dataset – containing data such as billing and invoices, accounts, resource allocation, etc. – often does not contain much data that would fall subject to regulatory restrictions, instead being more of interest to rival companies. First-order and second-order masks are fairly easy to identify, and cover a large part of the sensitive information available, but third-order masks are much harder to design as there are a great many different possible aggregations that could be beneficial to a rival company. As such, applying first-order and second-order masking to the dataset can produce quite a drop in crackability after quite a short time, but after that the third-order masking is a comparatively slow process.

An HCM system, containing payroll and HR information, is very heavily subject to regulatory restrictions, and so more care must be taken when masking to ensure compliance. The data to be masked is of a more complex format; things like names and addresses, as opposed to simple numbers, therefore the first- and second-order masks, while quite easy to identify, take longer to design and apply. These data sets have a very high demand for third-order masks, however: it is the nature of HCM systems that outliers are common, as there are a great many factors differentiating each person in the system from another. These outliers are often identified through crossreferenced records from multiple tables, making them much harder to identify. The end result is a slow but steady first-order and second-order masking process, followed by a third-order process that quickly slows down once the most obvious masks have been implemented.

For comparison purposes, we also included a synthesized dataset on the graph. A synthetic dataset naturally begins in such a way that crackability is low – as most of the information in the production dataset was never anywhere near the test dataset. A little effort may be required to ensure that the generated data does not accidentally contain any ‘real’ data, for example ensuring that none of the synthesized customer names actually match those of real customers, but quickly the data reaches a level of crackability such that the only information remaining is that which has been left in deliberately, such as trends in the data.

Conclusion

Most projects do not realise the possibilities when it comes to drawing real-world conclusions from masked data. The masking that can be most easily applied – first-order and second-order – still exposes a very large amount of longer-term business intelligence in the form of trends and structures. Attempting to mask data at this level is slow and requires a lot of creativity; most notably, it gambles that the team implementing the data masking has the imagination to think of all the possible things that any adversaries might wish to use against them. Total uncrackability is impossible without completely invalidating the testing process, but data synthesis can restrict the information available from the data to only those pieces of information that have been explicitly and deliberately included.

About Grid-Tools

Grid-Tools are specialists in test data creation, data masking and test data management. Their experienced personnel have been writing and developing solutions for large companies in both the private and public sectors for over 30 years.

The Grid-Tools Datamaker suite includes a wide range of tools for test data management including such innovative products as Datamaker, a revolutionary tool that creates and publishes quality test data from scratch whilst keeping the relationships and referential integrity of production environments. An invaluable tool for testing and development, Datamaker places the data into a central data repository so it can be used, inherited, manipulated and re-used across an entire organization. Voted “Most Innovative Testing Tool of 2008” by QA Guild, the functionality Datamaker provides has proven to be innovative, efficient and different than any other test data management product in the market. You simply will not find another tool like it!

Grid Tools

11 Oasis Business Park
Eynsham
Oxfordshire
OX29 4TP

UK: +44 01865 884 600

US: +1 866 563 3120

E: info@grid-tools.com

www.grid-tools.com

Find us on Facebook

Follow us on Twitter

Join the Datamaker circle on LinkedIn

Subscribe to our blog



DATA FIT FOR PURPOSE