What Else is Revealed by Order-Revealing Encryption?

F. Betül Durak
Rutgers University

Thomas M. DuBuisson
Galois, Inc.

David Cash
Rutgers University

ABSTRACT

The security of order-revealing encryption (ORE) has been unclear since its invention. Dataset characteristics for which ORE is especially insecure have been identified, such as small message spaces and low-entropy distributions. On the other hand, properties like one-wayness on uniformly-distributed datasets have been proved for ORE constructions.

This work shows that more plaintext information can be extracted from ORE ciphertexts than was previously thought. We identify two issues: First, we show that when multiple columns of correlated data are encrypted with ORE, attacks can use the encrypted columns together to reveal more information than prior attacks could extract from the columns individually. Second, we apply known attacks, and develop new attacks, to show that the *leakage* of concrete ORE schemes on non-uniform data leads to more accurate plaintext recovery than is suggested by the security theorems which only dealt with uniform inputs.

Keywords

database encryption; order-revealing encryption; inference attacks

1. INTRODUCTION

Property-preserving encryption (PPE) encrypts data so that certain functions of plaintexts are computable from ciphertexts by someone without the key. PPE flavors include *searchable encryption* [15] for text searching, *deterministic encryption* (where one can detect if two plaintexts are equal), and *order-revealing encryption (ORE)* [2, 4], where one can detect the order of two plaintexts without decrypting. PPE generally achieves weaker security than traditional encryption because it inherently leaks information about plaintexts. On the other hand PPE has enabled efficient encrypted database applications [14, 8, 1, 3].

This work considers the security of ORE. An ORE scheme is an encryption algorithm $\mathcal{E}$ that takes numbers from some domain as input, and such that, given two ciphertexts $\mathcal{E}_K(x)$, $\mathcal{E}_K(y)$, anyone can tell if $x < y$ without knowing K. ORE is useful for encrypting columns in a database table whilst still allowing range queries: To issue a query for the range (a, b) one encrypts a and b and sends the ciphertexts. The server can then find all rows with values between the encrypted endpoints in sublinear time.

ORE security. Starting with the work of Boldyreva et al. [4], ORE schemes have been proved secure with respect to mostly incomparable definitions. The strongest, which we will call *ideal* ORE, requires that only the order of ciphertexts can be learned by the adversary. Known ideal ORE constructions require heavy theoretical tools [6] or interactive protocols [14, 10, 9]. Other work [4, 5, 7] gave efficient, non-interactive constructions that achieve weaker security. Specifically, ideal security is relaxed to require that an adversary learn only the plaintext information contained in a *leakage profile*, which varies according to the construction.

Theoretical properties, like one-wayness, were identified for some definitions and used to compare leakage profiles. The proofs of one-wayness for ORE assume that data are uniform over some space, but this does not appear to be the case in any application we could think of. Thus it is not clear what the proofs imply about ORE in practice.

Further complicating the situation is the nature of the leakage profiles, which (depending on the construction) includes plaintext bits or statistics about plaintexts. The behaviour of leakage profiles on non-uniform data has not been explored either empirically or theoretically, making it difficult for practitioners to make informed decisions regarding deploying and configuring ORE.

A recent work of Naveed, Kamara, and Wright [12] initiated the empirical study of ideal ORE security via message recovery attacks against columns of real medical databases encrypted with ORE. Their attacks apply to any ORE that reveals the order and equality of plaintexts, including ideal ORE and any of the less-secure versions. They showed that ORE constructions are insecure when the entire plaintext space of a column is encrypted, and also gave attacks that used frequency statistics of plaintexts (from training data) to guess plaintext messages effectively. Moreover, they found that several columns of medical data were susceptible to their attacks.

The Naveed et al. attacks did not take advantage of the additional leakage that is present in all efficient constructions. Thus, their work gives an upper bound on the security of ORE, but leaves open the possibility that stronger attacks exist against the weaker constructions.

CCS '16, October 24–28, 2016, Vienna, Austria.

ISBN 978-1-4503-4139-4/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2976749.2978379

1.1 Our contributions

This paper presents a collection of observations, experiments, and attacks dealing with the use of ORE on data that will be encountered in practice. We point out overlooked properties of leakage profiles, perform experiments to measure the security of ORE against known attacks but on non-uniform data, and give new attacks showing that ORE is less secure than theory indicated.

Inter-column correlation-based attacks. In a database table with multiple columns, PPE-based systems [14, 8] use ORE to encrypt each column independently under different keys. Using a different key for each column prevents direct comparison of ciphertexts across columns, thus leaking less information.

The first part of this work investigates the security of ideal ORE on multiple columns. We observe that columns of data in a table are usually *correlated* because a row of a table usually corresponds to an individual record. This opens up a new avenue for attack, where an attacker attempts to extract information from multiple columns simultaneously rather than from the columns individually.

We study the effect of correlation using simple multi-column versions of attacks by [12] on ideal ORE. We analyze our attacks using visualizations and measurements of the attack accuracy on geographic datasets (described below) where latitude and longitude were correlated. We show that the rough shape of the geographic distributions is present in the ciphertext leakage, and also measure how accurately the plaintexts can be estimated by an adversary who knows a bounding box for the plaintexts.

In the course of our analysis, we advocate for a more general interpretation of attack success: Whereas Naveed et al. measured the fraction of plaintexts recovered exactly, we instead consider approximate recovery of plaintexts. This also allows us to consider datasets where values never repeat, meaning they have trivial frequency information that is not useful for the attacks of [12].

Evaluating concrete leakage profiles. We observed that when data in a column are not uniformly random and independent, the one-wayness proofs by [5, 7] do not apply. Thus, in practice, the information leaked by these constructions may be greater than the theorems suggest.

We show that non-uniformity of data does more than technically violate the assumptions of the theorems. We start by measuring the amount of information that can be directly inferred from leakage profiles of existing ORE constructions on different types of data, including synthetic and real location datasets and timestamps for mobile phone usage. This measurement consists of running simple attacks, and in some cases composing them, to produce guesses of plaintexts which are then evaluated for accuracy depending on the context.

We found that the security gap predicted by one-wayness theorems was not always present on real data. Concretely, we consider the construction of Chenette et al. [7], which was proved to be one-way in a quantitatively stronger sense than the prior work of Boldyreva et al. [4, 5]. But by simulating the Chenette et al. and Boldyreva et al. leakage profiles on a real dataset of 2000 latitude-longitude pairs, we found that essentially the same number of plaintext bits (about 50%) were explicitly leaked by both schemes. On larger datasets Chenette et al. may leak even more.

We took a closer look at the Boldyreva et al. scheme [5]. Theoretical results [4] suggested that this construction "leaks the most-significant half of the plaintext bits," but our experiments showed that this conclusion was too generous: Simple attacks recover more. Specifically, that construction appears to leak all of the leading zeros or ones of a message, *plus* the most significant half of the bits after the leading one (or zero). So, for example, an encryption of the zero message can usually be recognized exactly. The contrast with the theoretical result is explained because messages with many leading zeros or ones are unlikely to occur at random.

We additionally noticed that the Boldyreva et al. scheme often leaks these plaintext bits in the clear. Previously it was observed [5] that these bits could be inferred via computation, but in fact, one can simply glance at ciphertexts.

Inter+Intra-column correlation-based attack. Our results above show that the Chenette et al. construction can leak the same number of bits as Boldyreve et al., despite the one-wayness gap. We go further in attacking the Chenette et al. construction, showing how to infer additional plaintext bits beyond those explicitly leaked. Our attack exploits both *inter-* and *intra-column* correlations. The attack uses the observation that location datasets tend to cluster, so after some bits are revealed, hidden bits can be guessed to fit this tendency.

Our attack against the Chenette et al. construction was able to predict almost every point in a location dataset of California road intersections to within 5km (and most to within 0.5km), while the leakage profile did not explicitly leak the location of any plaintext to within less than 400 km. We note that our quantitative claim depends on the attacker determining the two most significant bits of the longitudes by hand, since the ORE construction did not leak these explicitly. (This amounts to putting a California-shaped blob of points in one of four possible places on the globe.)

We performed a similar experiment with timestamps of an individual's mobile phone usage, showing that most of the periods of time between usages could be estimated to within one hour.

Chenette et al. recommended that their construction be combined with the Boldyreva et al. construction. We show that our attack performs essentially as well against this version, calling into question whether the increased complexity and computation of a combined scheme is justified.

Additional observations. We revisit the security provided by *modular ORE (MORE)*, an enhancement to ORE suggested by Boldyreva et al. [5], and developed further by Mavroforakis et al. [11]. MORE applies an additive mask to the input before encrypting, which provably hides information when messages are uniform. But, when the data are not uniform, the proof does not apply. In fact, we show that the additive mask is likely easy to remove in practice, by visualizing MORE applied to the geographic data used by Mavroforakis et al.

Conclusions and recommendations. This work shows that the conclusions of one-wayness theorems about ORE security should not be assumed to hold on real data. But in practice we expect attacks to recover even more information than our experiments did. Our datasets are relatively small, and leakage gets worse as the dataset grows. In this paper, we highlight the performance of attacks on specific datasets, but these numbers should not be taken as "typical"

numbers to inform deployment decisions. This is especially true because the relationship between the semantics of data and its encoding will affect the attacks. Instead, we recommend that one run our attacks, which are all simple to implement and ran in a few minutes on our datasets, on test data (similar to production data for the intended use-case) before using ORE.

Some of the work in this paper grew out of exactly this approach, where the authors were considering ORE to store personal location histories. While we could not quantify the attacks theoretically, the plots in Figure 3 convinced us that ORE provide inadequate security, especially against adversaries with side information on the individual.

Paper organization. In the remainder of this section we review related work. In Section 2, we recall notation, security definitions, and prior attacks on ORE. In Sections 3 and 4, we present our attacks on ideal and leaky-ORE respectively.

Related work. ORE was first introduced by Agrawal et al. [2] and formalized by Boldreva et al. [4, 5] as a symmetric, deterministic, and stateless encryption scheme. Other works [14, 10, 9] constructed interactive protocols which achieve a functionality similar to ORE, whilst only leaking order (or less). Recently [6] constructed ideal ORE using multilinear maps, and their constructions is much less efficient than non-ideal, blockcipher-based, constructions.

In a recent work, Naveed et al.[12] explored *inference attacks* on PPE-based EDB systems. Portions of their work target the ORE- and deterministically encrypted columns, and we review the ORE-relevant part of their work below.

2. PRELIMINARIES

Dataset formalization and notation. We formalize a column as a vector $\mathbf{d}$ over some ordered plaintext space. Depending on the context, the plaintext space will be a set of fixed-length bitstrings, or set of positive integers $\{0, 1, \ldots, M\}$.

Given a bitstring $x \in \{0,1\}^n$ and positive integer k, we define $x \gg k$ to be the right shift of x by k bits, where the leftmost bits are not padded (so $0110 \gg 2$ is 01).

2.1 Order-revealing encryption

An ORE scheme consists of three algorithms $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{C})$ for key generation, encryption, and comparison respectively. The key generation algorithm outputs a key k, and the encryption algorithm may be randomized, takes an input message m from an associated ordered plaintext space, and emits a ciphertext c. The comparison algorithm takes two ciphertexts c_0, c_1 as input and outputs a bit b indicating that the message in c_b is larger (or $\perp$ if the messages are equal). When the algorithm $\mathcal{C}$ is a canonical numerical comparison operator, the scheme is called an *order preserving encryption (OPE)* scheme, though this distinction is not relevant for our attacks.

Security of ORE. We follow Chenette et al. [7] in defining ORE security with a parameter called a *leakage profile* $\mathcal{L}$. Formally, $\mathcal{L}$ can be any function on vectors of messages. Intuitively, an ORE scheme must leak the order of the plaintexts, but it may also be allowed to leak more, namely $\mathcal{L}$ applied to the messages. The formal definition requires that an adversary cannot win a game requiring it to compute more information than is output by $\mathcal{L}$. We proceed informally since our attacks do not depend on the details of the definition, but rather only on the leakage profile.

Five leakage profiles have been considered in the literature. We term them $\mathbf{Ideal}, \mathbf{ROPF}, \mathbf{MSDB}, \mathbf{RtM}, \mathbf{MtR}$. The first two were introduced by Boldyreva et al. [4], who construct an **ROPF**-secure ORE. Later, Boldyreva et al. [5] proved that **ROPF**-security requires an ORE to roughly reveal half the plaintext bits. Later, Chenette et al [7] defined profiles $\mathbf{MSDB}, \mathbf{RtM}, \mathbf{MtR}$, built ORE achieving them, and proved a result showing that these profiles leak fewer bits on uniform data. We review them in more detail now.

Ideal. The *ideal* leakage profile only reveals the order of the ciphertexts. This profile for instance hides any statistical information about the gaps between messages. The profile is achievable using (currently impractical) theoretical tools [6]. Ideal security is also achieved by *interactive* variants of ORE where encryption is a protocol between two parties [13, 10, 9]. We consider these protocols in-scope for this work, and will treat them as ORE. We note that [9] actually achieves stronger-than-ideal security by hiding the frequency of plaintexts in the column. Some of our attacks will still apply to this construction.

ROPF. The *random order-preserving function* profile [4] is defined with respect to a plaintext space and range. Given a column of data $\mathbf{d}$ with n cells, the leakage profile chooses a *random order-preserving function* f from the plaintext space to the range, and outputs $f(\mathbf{d}[1]), \ldots, f(\mathbf{d}[n])$, i.e. the function applied component-wise to the dataset.

The **ROPF** profile was later shown to reveal approximately the most-significant half of the plaintext bits [5] of a random message, and also to hide roughly the other half. The selection of the range set is a parameter to be set when configuring the instantiation.

MSDB. The *most-significant-differing bit* profile [7], on a column $\mathbf{d}$ with n entries, will output the order of the plaintexts in $\mathbf{d}$ along with, for all $1 \le i < j \le n$, a number $\mathrm{diff}_{i,j}$ that indicates the index of the most significant bit where the plaintexts $\mathbf{d}[i]$ and $\mathbf{d}[j]$ differ, along with their bits at that position. Equivalently, $\mathrm{diff}_{i,j}$ is the length of the longest common prefix of $\mathbf{d}[i]$ and $\mathbf{d}[j]$, plus one.

For example, if the plaintexts are $0000, 0001, 1000$, the **MSDB** profile would allow one to infer the first bit of all three plaintexts, and the last bit of the first two plaintexts, along with equality of appropriate prefixes. For this example, an adversary would learn that the plaintexts must be of the form $0uw0, 0uw1, 1xyz$, where u, w, x, y, z are variables for bits that are not explicitly leaked.

RtM and MtR. We consider two more profiles that are induced by composing multiple ORE schemes as suggested by Chenette et al. [7]. The first, **RtM**, is induced by first applying the leakage profile **ROPF** to get a vector $(f(\mathbf{d}[1]), \ldots, f(\mathbf{d}(n)))$, and then applying **MSDB** to this vector (treating it as if it were a plaintext). That is, the profile leaks the index of the most significant differing bit of $f(\mathbf{d}[i])$ and $f(\mathbf{d}[j])$ for each $i < j$.

The profile **MtR** will be scheme-dependent. The scheme is defined by composing some **MSDB**-secure OPE scheme with an **ROPF**-secure ORE scheme. The leakage is defined by the output of the composed scheme. Note that we must assume that the **MSDB** scheme here is OPE, not just ORE, in order to define the compare algorithm for this version.

The two profiles **MtR**, **RtM** were originally introduced without distinction, but we observe that they provide formally different security.

2.2 Theoretical results on leakage profiles

One-wayness of ROPF. Boldyreva et al. [5] introduced the notion of *window-one-wayness (WOW)* for ORE. An ORE is L-WOW if given $\mathcal{E}_K(x_i)$ for several uniformly random strings $x_i \in \{0,1\}^m$, it is infeasible to determine an interval of size L containing some x_i. To make our experiments easier to compare, we consider an alternative version that we call ℓ-bit-WOW, which says that it should be infeasible to compute the ℓ-bit prefix of some x_i.

Boldyreva et al. [5] also proved that any **ROPF**-secure ORE with plaintext space $\{0,1\}^m$ is L-WOW secure for L approximately $2^{m/2}$, meaning that it is hard to compute the lower $m/2$ bits of a uniformly random plaintext.

One-wayness of MSDB. Chenette et al. [7] proved that **MSDB**-secure OREs have stronger WOW security. That work proved ℓ-bit-WOW security for **MSDB** schemes, for a much smaller ℓ of about $\ell \approx 1/\log \epsilon$, where ϵ is the desired bound on the success probability of the adversary. (A smaller ℓ means the result is stronger, and it is hard to guess even a smaller prefix of the plaintext.) A closely matching attack against ℓ-bit-WOW of **MSDB** is almost immediate, as one can show that the **MSDB** leakage profile will provide roughly this many bits of a plaintext prefix.

Security of RtM and MtR. Composed ORE schemes were suggested in [7] to combine the security of **ROPF** and **MSDB**. It was proved that the composition of individual OREs will result in a construction that inherits both security notions. We remark that this is only a lower bound on the quality of security achieved, and that the composition may be strictly more secure than **ROPF** or **MSDB**.

2.3 Attacks on leakage profiles

Scaling attack against ROPF. Boldyreva et al. [5] introduced what we term the *scaling attack* and denote ScalingAtk against the WOW security of an **ROPF**. It works as follows. To attack an ORE with plaintext space $\{0,\ldots,M\}$ and range $\{0,\ldots,N\}$, one maps a ciphertext y for an unknown plaintext x to $x' = \lceil yM/(N+1) \rceil$. It was proved that this estimate will satisfy $|x - x'| < 8\sqrt{M}$ with high probability.

We observe that when $M = 2^m$ and $N = 2^n$ are powers of two, this can be approximated by a simple bitshift: $x' \approx y \gg (n-m)$ i.e. the right-shift of y until it is the same bit-length as a plaintext. The estimate for these parameters becomes $|x - x'| < 2^{m/2+3}$, now implies that for most x, about the upper half of the bits of x' will match those of x. We omit a formal analysis of the variation due to high-order bit rollovers.

Sort attack against Ideal. Recent work by Naveed et al. gave attacks on **Ideal**-secure ORE schemes (and thus on all of the other profiles except that of [9]). The first was the sort attack, denoted SortAtk below, which we recall in detail for use later. The attack assumes knowledge of a plaintext space that we identify with $\{1,2,\ldots,M\}$, and attempts to guess the plaintexts used to generate a vector of ciphertexts $\mathbf{c}$ that it takes as input. The attack sorts the vector $\mathbf{c}$ (using the ORE comparison algorithm), and then guesses that the smallest unique ciphertext corresponds to 1, that the second smallest ciphertext corresponds to 2, and so on. Formally, it is defined as follows. Let $c_1, c_2, \ldots$ be the ciphertexts in $\mathbf{c}$ in sorted order. $\mathsf{SortAtk}(\mathbf{c})$ outputs a mapping α from ciphertexts to $\{1,\ldots,M\}$, where

$$\alpha(c) = \begin{cases} i & \text{if } c \in \mathbf{c},\ c = c_i \\ \perp & \text{otherwise} \end{cases} .$$

The sort attack was shown to correctly invert a large fraction of ciphertexts in a simulated attack. However, it required that the plaintexts were "dense" in the message space, meaning that almost all possible plaintexts in $\{1,\ldots,M\}$ are encrypted in $\mathbf{c}$. This was true for several columns in certain hospital databases, like age (years), length-of-stay (0-365), and others.

Cumulative attack against Ideal. Naveed et al. gave a second attack, called the cumulative attack and denoted CumulativeAtk that works when plaintexts are not dense in the message space. This attack also takes as input a vector of ciphertexts $\mathbf{c}$, but additionally requires a training vector $\mathbf{z}$ of data which should be drawn from the same distribution as the plaintexts in $\mathbf{c}$.

The cumulative attack outputs a map α from ciphertexts in $\mathbf{c}$ to plaintexts in $\mathbf{z}$. The map α is computed via a linear program that minimizes the error in frequencies and in the cumulative distribution (i.e., the fraction of plaintexts less than a given plaintext in $\mathbf{z}$ versus $\mathbf{c}$). We omit further details, but we note that the training input $\mathbf{z}$ is essential for two reasons: First, the map α will only output plaintexts in $\mathbf{z}$, and thus if a plaintext has not been seen, then $\mathsf{CumulativeAtk}(\mathbf{c}, \mathbf{z})$ will never guess it. Second, the guesses are entirely dependent on using $\mathbf{z}$ as a "typical" distribution with frequencies that correspond to the target data.

Other attacks. It was observed by Boldreva et al. [4] that *chosen plaintext attacks* allowed the easy extraction of plaintexts. This, and attacks that observe queries, are not considered in this work but are likely to further diminish security in practice.

2.4 Datasets and implementation

We use two real geographic datasets `Cal`, `SpitzLoc`, one synthetic geographic distribution `Globe`, and one real timestamp dataset `SpitzTime`.

The dataset `Cal` represents the latitude and longitude of about 21,000 intersections in the California road network[1] (also used by Mavroforakis et al. [11] in their ORE work). Latitudes are numbers between -90 and 90 and longitude are numbers between -180 and 180, both given to six decimal digits. Encoding latitude requires $\lceil \log_2(180) \rceil = 28$ bits, and similarly longitude requires 29 bits. We encode a given latitude x as $10^6(x+90)/180$ in binary, and a longitude y as $10^6(x+180)/360$ in binary. The latitudes were all between 32.541302 and 42.017231, and longitudes were between -124.389343 and -114.294258.

The dataset `SpitzLoc` consists of latitude and longitude coordinates tracking the movement of German Green party politician Malte Spitz over six months. The dataset records the location of towers used for voice calls and text messages, as well as times (which we used for the `SpitzTime` dataset

[1]Dataset obtained from http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm

below). The data were provided by Deutsche Telecom and posted publicly as an example of highly sensitive information recorded by service providers[2]. The dataset consists of about 36,000 rows, with many locations missing. We extracted from this 1,477 non-repeating (latitude, longitude) pairs in Germany that we call `SpitzLoc`. The points were available to 8 decimal digits of accuracy, but we encoded them as 28 and 29 bit-long strings as in `Cal`.

The `SpitzTime` dataset consists of 30,492 time-stamps represented as seconds from an epoch of January 1, 2000. The actual range is between 2009-08-31 to 2010-02-27 and mostly daylight hours. Concretely, the timestamps were integer values between 305,020,620 and 320,629,560.

Finally, we selected a distribution `Globe` to represent the latitude and longitude of a uniformly random point on Earth. We encode the points as bitstrings of length 32, using the same method as before. We chose this distribution because it does not result in uniform samples of pairs from $\{0,1\}^{32}$, but it may model the distribution of some datasets.

All experiments were performed on recent Mac laptops. Our experiments were written in Python and used the **ROPF** implementation from CryptDB. The other leakage profiles could be simulated exactly without full implementations.

3. INTER-COLUMN CORRELATION

The sort- and cumulative-attacks [12] showed that the ciphertexts (produced with **Ideal**-secure ORE that only leaks order and frequency) in an individual encrypted column could sometimes be inverted, but required at least one of the following conditions to be met:

- The plaintext data present in the column is *dense* in the plaintext space, meaning that most or all of the possible plaintext values appear at least once.
- The plaintext data has *low entropy*, meaning most of the possible plaintexts appear frequently, and particularly that a training set will have many plaintexts in common with the column under attack.

We add to this another condition: When two or more encrypted columns hold data that are *correlated*. We show that information may be leaked even when the data in the column is sparse in its domain, and when all values are unique (and without any training data), and an **Ideal**-secure ORE is used. We call this *inter-column correlation*, and below we experiment with applying the sorting attack on multiple columns at once.

2-D sort attack. We consider applying the sort attack to two columns at once. Recall that, given a ciphertext vector $\mathbf{c}$, $\mathsf{SortAtk}(\mathbf{c})$ outputs a mapping α from ciphertexts to the set $\{1,\ldots,M\}$, where M is the number of unique ciphertexts in $\mathbf{c}$.

We simply apply this attack to two columns individually. That is, we define the attack $\mathsf{2DimSortAtk}(\mathbf{c}_1,\mathbf{c}_2)$ to output (α_1,α_2) given by $\alpha_i \leftarrow \mathsf{SortAtk}(\mathbf{c}_i)$ for $i=1,2$. In words, this attack is independently sorting each ciphertext vector, and emitting guesses about the plaintext vectors using the same technique as $\mathsf{SortAtk}$. Below, we argue that this attack should be interpreted differently from the original single-column attack.

[2]The data is downloadable at http://www.zeit.de/datenschutz/malte-spitz-data-retention

Visual example. We start with an example. In Figure 1 (a), we plot an image, which is formally a set of approximately 10^6 points in $\{1,\ldots,2000\}^2$ (the black points correspond to points in the set). In (b), we select a random subset of only 300 points from the set, conditioned on none of the coordinates repeating (that is, none of the chosen points have the same x or the same y value).

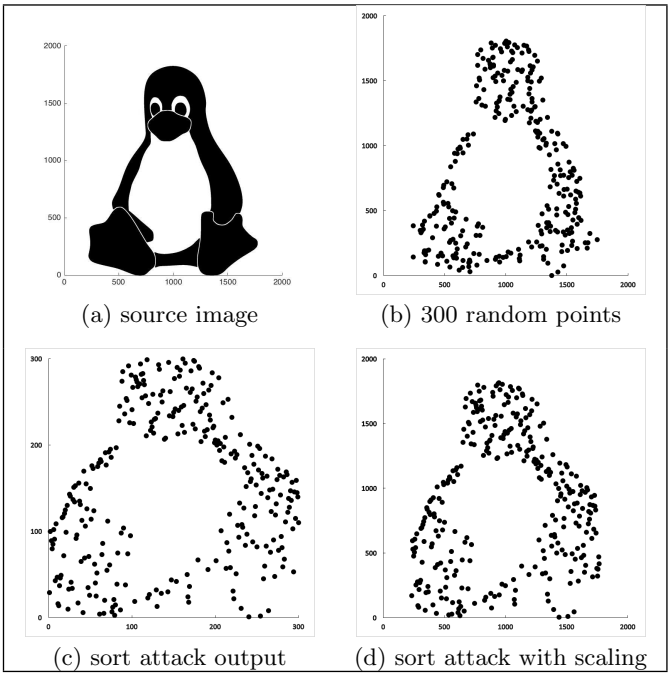


Figure 1: Visualization of the 2-D sort attack on an image dataset encrypted with **Ideal** ORE.

We model the chosen subset as a dataset $\mathbf{d} = (\mathbf{d}_x, \mathbf{d}_y)$ consisting of a pair of columns $\mathbf{d}_x$ and $\mathbf{d}_y$ each with 300 cells, where each row represents the location of a black point in the image. We selected the data in this way to ensure that prior attacks against the two individual columns would not be effective. Namely, since all points in each column are unique, the frequency information is trivial. Moreover, each column only consists of 300 out of 2000 possible plaintext values, so the columns are not dense in the plaintext space.

Despite the prior attacks failing to recover the plaintexts in $\mathbf{d}$, we show that information can be recovered by considering the columns together via the 2-D sort attack. The results of our attack are plotted in (c), where the rough features of $\mathbf{d}$ are still visible.

Our multi-dimensional sorting attack does not guess any points correctly – The original points are in $\{1,\ldots,2000\}$ but the sort attack only emits guesses in $\{1,\ldots,300\}$. Thus, by the metrics of [12], the attack does not work. But it is clear that much structure of the image is recovered, including relatively fine details like the arrangement of points from the penguin's foot in the lower right. (By scaling the plot in (d) it resembles the plaintext points more strongly.)

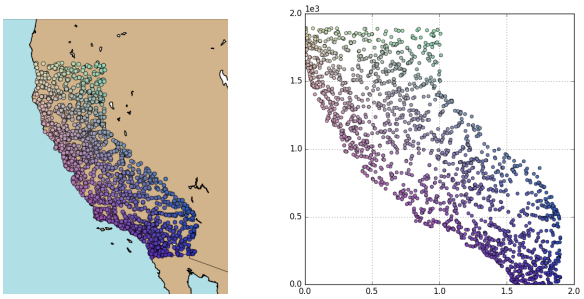
This experiment suggests that we expand our consideration of the leakage of ORE in two senses. First, even when individual encrypted columns are useless for an adversary, the leakage from *correlated* encrypted columns may combine to reveal a harmful level of information, even to an adver-

sary with no training data to help its analysis. Second, even when an attack does not recover plaintexts correctly, it may still be plausibly considered successful by recovering partial information.

3.1 Sort attack on location datasets

We experimented with the 2-dimensional sort attack on two location datasets: `Cal` and `SpitzLoc` (see Section 2.4 for details).

`Cal` dataset visualization. For the `Cal` dataset, we selected a subset of 2,000 random points and ran 2DimSortAtk on the ideal leakage for that subset. In Figure 2, we plot the plaintext points and the output of 2DimSortAtk (the colors are used to track plaintexts between the two plots). We observe that the shape of California is still clearly visible, and some features like western protrusion one third of the way up are also visible (several other runs on random points were similar). Below we return to this attack and try to evaluate it quantitatively.



(a) 2,000 plaintext points (b) 2DimSortAtk output

Figure 2: Visualization of 2DimSortAtk on a subset of the `Cal` dataset.

`SpitzLoc` dataset visualization. We applied 2DimSortAtk to the entire Spitz location dataset, also keeping track of time ordering for visualization (we did not attempt to guess times in this part). This attack generates guesses for every location in the dataset, and we plotted subsets of the guesses corresponding to different periods of time in Figure 3. We selected a specific day, week, and month to examine and juxtaposed the guesses with the corresponding subset of the plaintext data. Consecutive guesses are connected by lines to recreate the movement history. We note that the plots are strikingly similar in some features, while some of the structure is lost. For instance, in the last row, movement to the North was compressed in the top of (b.3) because fewer points were reported in that region.

Conclusions. The authors initially investigated the security of ORE for two projects. The first required encrypting a database that included locations of naval vessels, and second considered encrypting personal mobile phone GPS histories to allow for location-based queries. Our experiments above showed that even the best possible leakage (**Ideal**) would still result in a concerning loss in secrecy.

We conjecture that the security in practice may be much worse. This is because we generated the plots above *without any training data or side information.* An attack who knows that `Cal` points are taken from California, or that `SpitzLoc` points are taken from the movements of a German citizen, can use side information like the shape of the movement zone and the distribution of cities and other points of interest when generating guesses at the plaintext data.

Moreover, our attacks are on relatively small amounts of data. For instance, a column with 2,000 rows could be downloaded and searched locally in most scenarios, meaning that ORE may be unnecessary there. On larger datasets, such as years of location movements, the guesses might be even more accurate.

3.2 Sort attack accuracy with bounds

We now consider extending the 2-D sort attack to use the additional hint of *bounds* on the possible plaintexts and generate guesses on the plaintexts that can be quantitatively evaluated. Concretely, we consider the following variant of 2DimSortAtk, denoted Bnd2DimSortAtk. In addition to the encrypted columns $(\mathbf{c}_1, \mathbf{c}_2)$, the attack also takes as input pairs of numbers (a_1, b_1) and (a_2, b_2). First, it runs the original 2DimSortAtk twice, to generate mappings (α_1, α_2) from the ciphertexts to $\{1, \ldots, M\}$. We then compose these mappings with functions f_1, f_2 that evenly space the guesses within the given bounds, resulting in the following attack:

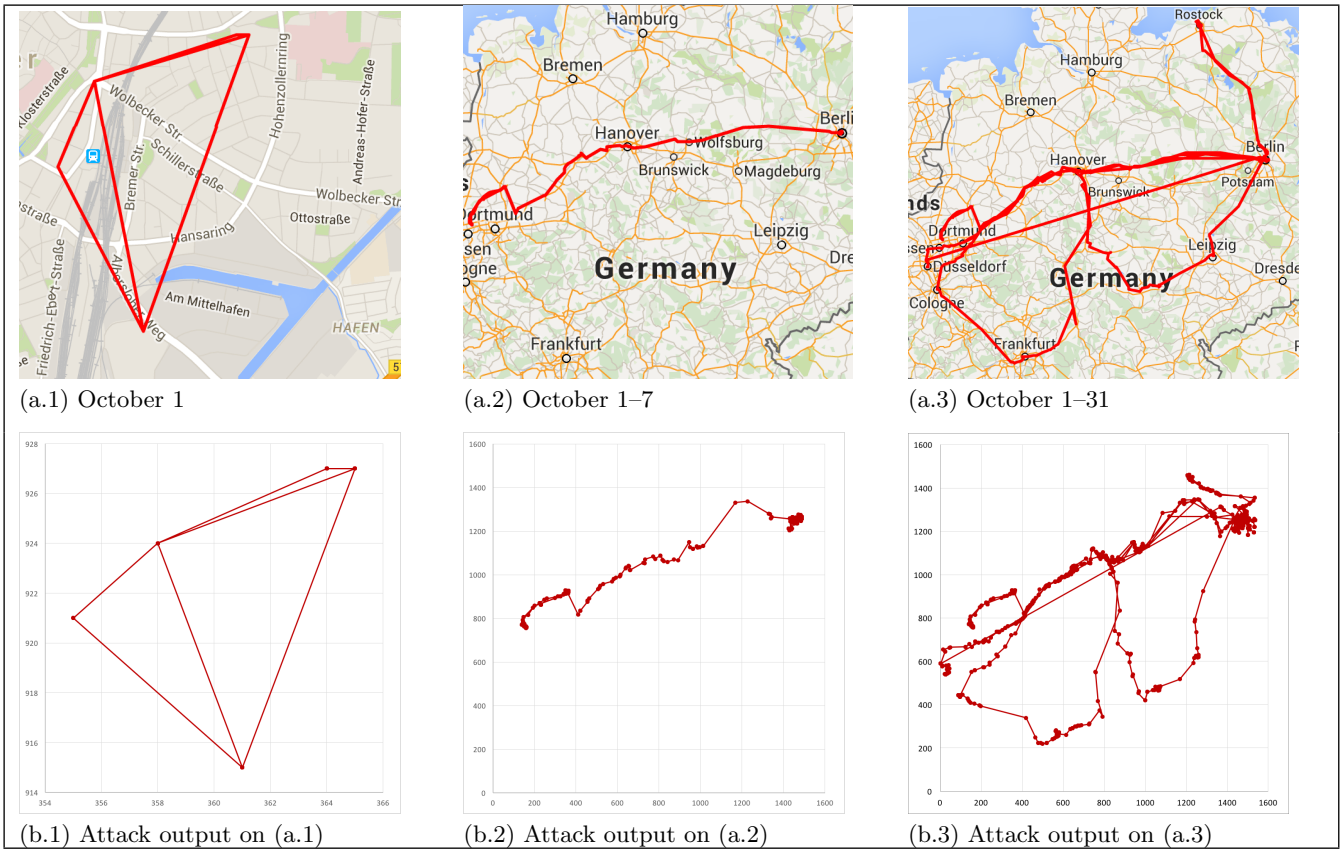
Bnd2DimSortAtk$(\mathbf{c}_1, \mathbf{c}_2, a_1, b_1, a_2, b_2)$
01 Compute $(\alpha_1, \alpha_2) \leftarrow$ 2DimSortAtk$(\mathbf{c}_1, \mathbf{c}_2)$
02 Define the function f_1 by $f_1(i) := (i-1) \cdot \frac{b_1 - a_1}{|\mathbf{c}_1|} + a_1$
03 Define the function f_2 by $f_2(i) := (i-1) \cdot \frac{b_2 - a_2}{|\mathbf{c}_2|} + a_2$
04 Output $(f_1 \circ \alpha_1, f_2 \circ \alpha_2)$.

In the algorithm, $|\mathbf{c}|$ denotes the number of ciphertexts in the encrypted column $|\mathbf{c}|$.

`Cal` dataset attack with bounds. We used random subsets of $25, 50, ..., 2000$ points from the `Cal` dataset to evaluate Bnd2DimSortAtk. The results are plotted in Figure 4. In each case we gave the attack the same bounds, which were set to the greatest and smallest latitudes/longitudes in California (and in particular, they were not the maxes and mins over the actual subset under attack). We plotted the quality of guesses as stacked histograms in Figure 4 (each bar reports on a different run, showing the proportion of points that were guessed to within different accuracies on that run).

The maximum error in any of the experiments was around 140 km while the minimum was about 2 km. We note that our plot reveals that the quality of guesses was not improving with the number of points, which is curious because we expected a dense set of points to reveal more ordering information. The explanation (which we found via inspection and plotting the guesses) is that bad guesses tend to stay bad, even with many points, and good guesses tend to stay good. We stress however that an attack, with, say, a training set of points in California could likely do much better than this simple attack.

Discussion. Strictly speaking, this attack is no longer exploiting inter-column correlation because Bnd2DimSortAtk could be adapted to run on individual columns, and it would generate the same guesses for the latitude and longitude columns independently. However we suggest it as a technique for evaluating the confidentiality of **Ideal** ORE on geographic data.



(a.1) October 1 (a.2) October 1–7 (a.3) October 1–31

(b.1) Attack output on (a.1) (b.2) Attack output on (a.2) (b.3) Attack output on (a.3)

Figure 3: Visualization of subsets of the 2DimSortAtk output on the `SpitzLoc` dataset.

4. LEAKAGE-ENABLED ATTACKS

This section contains evaluations of known attacks on ORE but with non-uniform data (sections 4.1, 4.2, 4.3), showing that in some cases leakage is much worse than on uniform data. Then new attacks are presented (sections 4.4, 4.5, 4.6), and we also present our observations regrading MORE (section 4.7).

4.1 MSDB on random globe points

We evaluated **MSDB** leakage on the `Globe` dataset. Recall that **MSDB** allows one to infer some bits explicitly but keeps others hidden. In order to quantitatively compare the leakage to the plaintext values, we filled in all unknown bits arbitrarily.

The accuracy of our guesses is evaluated in Figure 5. The **MSDB** leakage profile tends to leak more information on larger datasets (it is *monotonic* in the sense that it will never leak less when a subset of data is included in a larger set). We note that after 600 points, we found that over half the points could be guessed within 10km, and the other half could be guessed within 50km. Thus even on a tiny dataset of random values, the geometric meaning of this leakage (which is limited mostly to high-order bits) reveals significant information about the dataset.

It is a fair observation that the `Globe` dataset is geographically uniform and numerically non-uniform, which undoubtedly impacts the quality of the attack. The same attack on numerically uniform data, which has a geographic distribution biased towards the poles, results in a slight improvement in mean attack accuracy without taking advantage of the known distribution. We did not analyze datasets with encodings that maintain numeric and geographic uniformity.

4.2 ROPF on real locations

Boldyreva et al. [5] proved that any **ROPF**-secure ORE with plaintext space $\{0,1\}^m$ is L-WOW (see Section 2.2) for $L \approx 2^{m/2}$. This roughly implies that it is hard to guess more than $m/2$ of the most significant bits of a random plaintext. Their result was involved, and generalizing it to other plaintext distributions does not seem easy. Moreover, real data may not obey and distribution assumed in a proof anyway.

Thus, we instead evaluated how the ScalingAtk (see Section 2.3) performed on our datasets. We encrypted subsets of `Cal` using an **ROPF**-secure ORE of [4], and recorded the length of the plaintext prefix that appears in the corresponding ciphertext. The average number of bits preserved between the plaintexts and corresponding ciphers over entire longitude column of `Cal` is 15 bits out of 27. This result matches the theory for uniform data. Moreover, Boldyreva et al. [5] also showed that increasing the ciphertext length by more than few bits beyond the plaintext length does not have any effect on the security for uniform data. We also experimented with varying ciphertext lengths, and the average number of preserved bits remained around 15 bits for larger output sizes.

We have no theoretical framework to explain how and if this experiment will generalize to other data. It seems

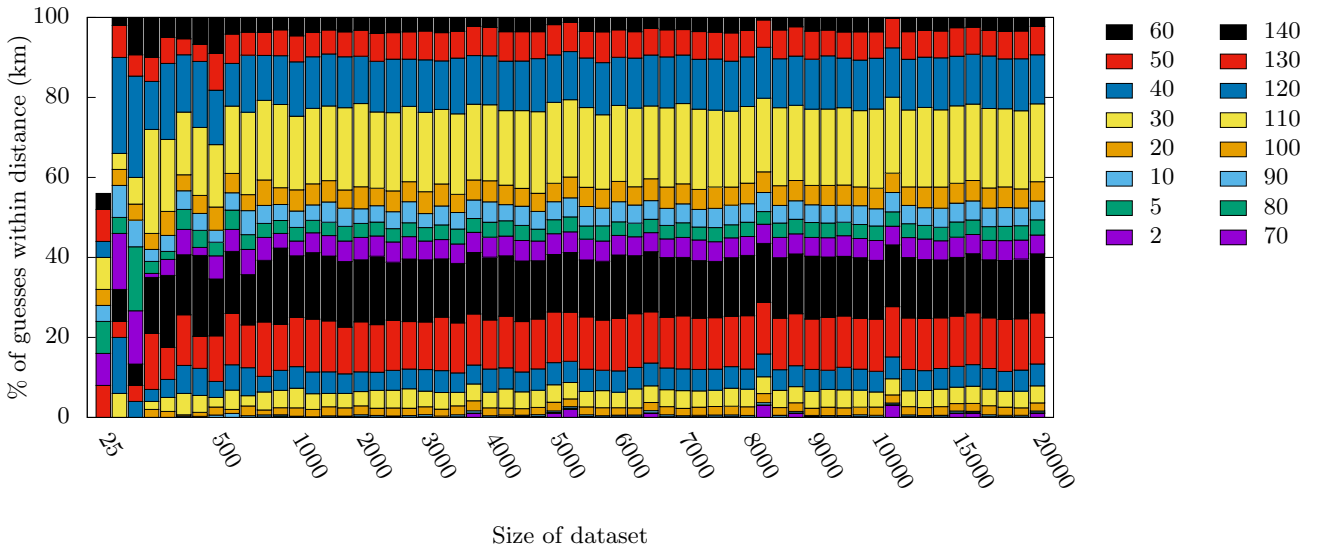


Figure 4: Histogram of 2DimSortAtk output on subsets of 25 up to 20,000 points (non-uniform step sizes). Distances in km from the correct plaintext point.

prudent (and easy) for practitioners to simulate this attack on test data before deployment.

4.3 ROPF on small and large messages

Boldreva et al. proved roughly that an **ROPF**-secure ORE will leak half of the bits of a random input message, but not much more, even when the output space is only one bit longer than the input space. Here we experimentally show that this result does not apply when encrypting messages that are close to the minimum and maximum elements of the message space. That is, for small and large messages, the Boldyreva et al. construction leaks far more.

We performed the following experiment. We fixed the message space of the Boldyreva et al. ORE to $\{0,1\}^{64}$, and output space to be either $\{0,1\}^{65}$ or $\{0,1\}^{128}$ in two independent runs. We encrypted the plaintexts $x = 2^0, 2^1, \ldots, 2^{63}$, to generate ciphertexts $c_0, c_1, \ldots, c_{63}$, and computed $x'_i \leftarrow \mathsf{ScalingAtk}(c_i)$ for each i. We computed the error $e_i \leftarrow |x_i - x'_i|$ for each i, and averaged the e_i over 10 independent runs (i.e. we selected a new key each time).

Note that the largest message in our experiment is $x = 2^{63}$, the midpoint of the message space. By symmetry, similar results would be obtained for the large messages near 2^{64}, so we did plot these results.

In Figure 6 we plot the logarithms (base 2) of the errors e_i compared to i, the logarithm (base 2) of the message. We find that the scaling attack performs much better on small messages than on random messages, which can be guessed to within a distance of about 2^{31}. The ciphertext for $x = 1$ was predicted exactly in every run. Ciphertexts up to 2^4 were recovered to with distance 2 on every run. The rest of the guesses were much more accurate than 2^{31}, until we reach larger messages. There was no significant variation when we changed the output length of the cipher.

Let n be the input length. The trend is that an input $x \approx 2^i$ or $x \approx 2^n - 2^i$ can be predicted to within about $2^{i/2}$ accuracy. This can be stated alternatively as a new rule of thumb: *An **ROPF**-secure ORE leaks* all *of the leading zeros (or ones) of a message, and* additionally *the most-significant half of the remaining bits.*

Discussion. The **ROPF** security result held because a random message was unlikely to be very small or very large. But it is easily possible that one will want to encrypt large or small values for application-dependent reasons. We suggest that the intuition about "leaking half the input bits" be updated, and that caution be exercised as the leakage may be even worse.

4.4 MSDB on real locations: The distance minimization attack

A stronger one-wayness result was proved for the **MSDB** profile. When random data are encrypted, the proof showed that only the k most significant bits will be leaked, except with probability about $|\mathbf{d}|/2^k$. Again, a detailed result on general distributions seems difficult to derive and anyway may not be useful in practice. Intuitively, the result follows because two random plaintexts will have the same k-bit prefix with probability $1/2^k$, and if this does not happen then no bits beyond the k-th will be leaked.

We evaluated this profile on the `Cal` dataset. Since points are not uniformly random, the location of differing bits between pairs will depend strongly on the distribution. Moreover, by exploiting properties of the distribution, even more bits may be inferred, as we show below.

Intuition. We first visualize **MSDB** leakage on the `Cal` dataset. Recall that this leakage profile explicitly leaks some bits of the plaintexts and keeps other bits hidden (see Section 2.1). In order to visualize the geometry of the leakage on `Cal`, we pretend that the unknown bits are the average of their possible values – that is, instead of one or zero, they are actually "0.5". After filling in these values we plot the result in Fig. 7. The large groups of points are separated from the main group when a relatively high-order bit is hid-

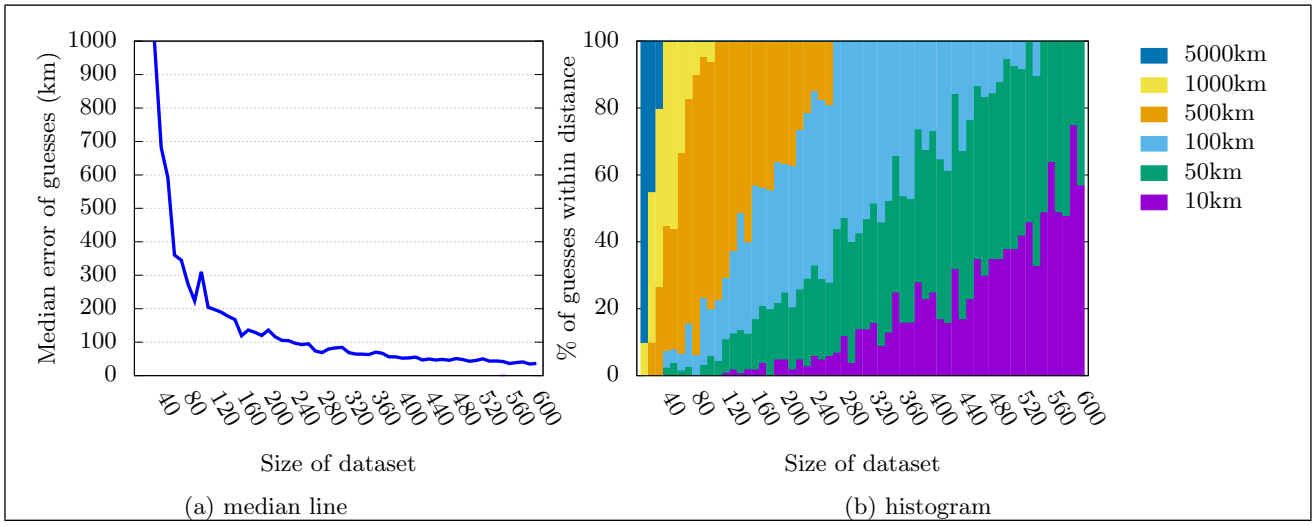


Figure 5: Accuracy for **MSDB** leakage on random globe points.

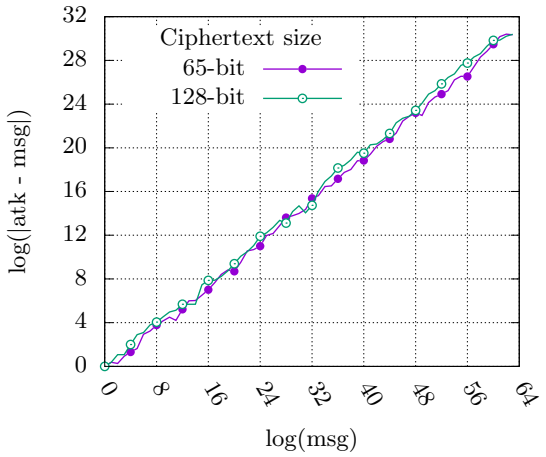


Figure 6: Accuracy of ScalingAtk against **ROPF** on small messages.

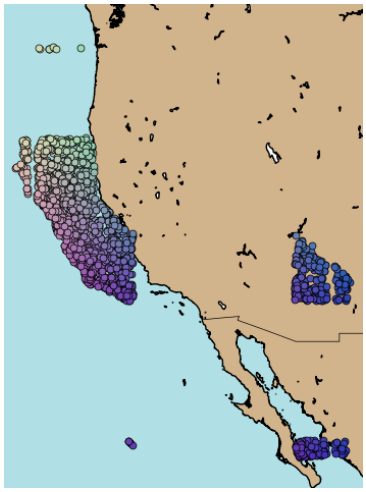


Figure 7: Visualization of **MSDB** leakage on `Cal` dataset.

den. There are also many other low-order bits hidden, but their effect is harder to see. Our intuition is that the large groups are trivial to move to the correct location by hand, and thus some of the hidden bits are easy to guess.

We give an attack to automate this, called the *distance minimization attack*, which is described and evaluated below. This attack will consider an individual encrypted column, and create guesses for every plaintext bit by moving the points to possible locations and seeing which is "closer" to the aggregate group, with the intuition being that correlated data will not often exhibit behavior like the irregular groups in Fig. 7.

The attack. The attack is given in pseudocode here, and a description follows.

DistMinAtk($\mathbf{c}$)

01 Initialize an empty guess vector $\mathbf{g}$
02 Foreach $\mathbf{c}[i] \in \mathbf{c}$
03 Set $\mathbf{g}[i] \in \{0, 1, \perp\}^m$ using **MSDB** leakage
04 Reset $\mathbf{g}[i] \in \{0, 1, 0.5\}^m$ by replacing $\perp$ with 0.5
05 For $j = m$ down to 1
06 Foreach $\mathbf{g}[i]$ with j-th bit unresolved (i.e. set to 0.5)
07 Try assigning j-th bit of $\mathbf{g}[i]$ to 0 and 1
08 For each setting, compute $S = \sum_k |\mathbf{g}[i] - \mathbf{g}[k]|$.
09 Set j-th bit of $\mathbf{g}[i]$ to minimize S.

The distance minimization attack is given a column $\mathbf{c}$ of ciphertexts encrypted with an **MSDB**-secure ORE. It first initializes a guess vector $\mathbf{g}$ using **MSDB** leakage naively (line 03). That is, it starts with all bits unknown. Then for every pair of ciphertexts $\mathbf{g}[i], \mathbf{g}[j]$, it performs the comparison to learn their differing bit, and then fills in that bit in the entries of $\mathbf{g}[i]$ and $\mathbf{g}[j]$.

After this initial stage, the guesses are strings in $\{0, 1, \perp\}^m$, where $\perp$ represents that a bit was not leaked. The rest of the algorithm will assign every $\perp$ entry to either zero or

one. The algorithm will need to interpret the guesses $\mathbf{g}[i]$ as numbers, even when it has unknown bits. We will temporarily set $\mathbf{g}[i]$ to the "average" of all the possible values that it could be. Concretely we convert each $\mathbf{g}[i]$ into a vector over $\{0, 1, 0.5\}$ by replacing $\perp$ with 0.5. Now when we need to consider $\mathbf{g}[i]$ as a number, we can compute its value using the binary expansion formula, but with values $0, 1, 0.5$ allowed as bits instead of just $0, 1$.

Starting on line 05, the attack begins to resolve the 0.5 entries in the guesses to either 0 or 1. It begins with the most significant bits, and considers the guesses with most significant bit unknown individually. When considering the guess $\mathbf{g}[i]$ the attacks tries setting the unknown bit of $\mathbf{g}[i]$ to 0 and 1. For each setting it measures the sum of absolute differences between the resulting point and all of the points in the guess set (it is at this point that we are considering the guesses as *numbers*, to compute differences). The attack selects the bit setting that results in a smaller sum, and moves on to the next guess.

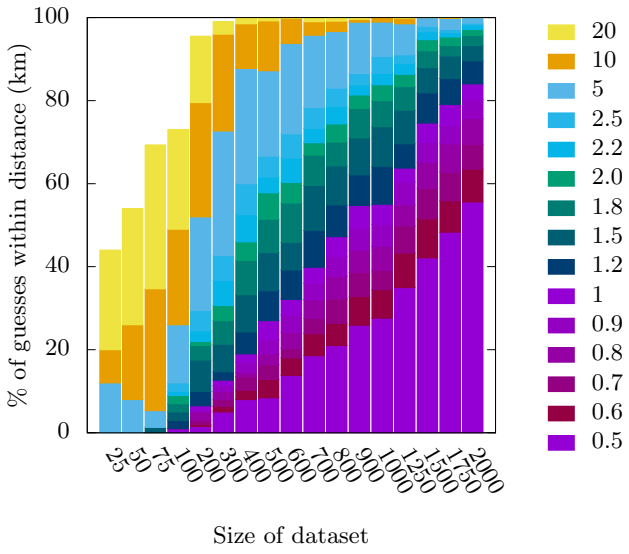


Figure 8: Accuracy of DistMinAtk against **MSDB** on `Cal`.

Evaluation. We evaluated this attack on random subsets of the `Cal` dataset of varying size. In `Cal`, the two most significant bits of longitude never change, so the attack cannot automatically infer these bits. This is exactly the sort of information that **MSDB** is designed to protect, but we assert that these can sometimes be guessed. In our setting, and attacker could run the attack without guessing these bits, notice that the shape of California appeared, and then fill in the missing significant bits by selecting one of four possible positions on the globe.

We evaluated the accuracy of our algorithm assuming the two most significant bits of longitude were given. In Fig. 8, we measure accuracy on subsets of sizes 25 to 2000. The figure reflects the improvement on the accuracy of guesses as the size of the dataset grows. On a dataset of 2,000 points, the algorithm guesses 95% of the plaintexts to within 2 km, and has average error 0.6 km. We note that no location was leaked to within less than 400km explicitly, and the improvement comes from the attack inferring hidden bits.

Discussion. For this particular example, leakage and correlation allowed accurate estimatation of essentially every plaintext. More generally, use of any ORE scheme on any dataset that contains correlations should be viewed with a healthy dose of caution. Limiting databases to small datasets might not offer meaningful protection and anyway negates the benefits of off-loading computation, which is a primary motivation for ORE. Practitioners desire simple rules for determining if property-preserving encryption and leaks inherent to the selected mechanism are acceptable, but no such rules have been proposed while examples of dangerous uses continue to mount.

4.5 Combined attacks on MtR and RtM.

In this section we consider how prior attacks can be combined to extract information from the composed leakage profiles **MtR** and **RtM** defined in Section 2.1. We start by describing how each can be attacked, and then evaluate the attacks.

In this section, let $\mathcal{E}^{\mathrm{r}}$ be the **ROPF**-secure ORE from [4] and $\mathcal{E}^{\mathrm{m}}$ be the **MSDB**-secure ORE from [7]. We will specify the appropriate domains and ranges for $\mathcal{E}^{\mathrm{r}}, \mathcal{E}^{\mathrm{m}}$ as needed below.

MtR. This profile describes the security of the following ORE scheme: It uses two random keys $K^{\mathrm{m}}, K^{\mathrm{r}}$. To encrypt a plaintext x, it computes

$$c_{\mathrm{in}} \leftarrow \mathcal{E}^{\mathrm{m}}_{K^{\mathrm{m}}}(x); \quad c_{\mathrm{out}} \leftarrow \mathcal{E}^{\mathrm{r}}_{K^{\mathrm{r}}}(c_{\mathrm{in}})$$

and outputs c_{out}. Note that we have assumed that the range of $\mathcal{E}^{\mathrm{m}}$ is contained in the domain of $\mathcal{E}^{\mathrm{r}}$. In order for the scheme to be correct (i.e., order-preserving) we need that $\mathcal{E}^{\mathrm{m}}$ is an *OPE* scheme, not just an *ORE* scheme, since the composition prevents running a general comparison algorithm on c_{in}.

Now suppose we are given a column $\mathbf{c}$ of ciphertexts encrypted with the above construction. We first apply the scaling attack to each ciphertext $\mathbf{c}[i]$ by running $\mathbf{y}[i] \leftarrow \mathsf{ScalingAtk}(\mathbf{c}[i])$ (see Section 2.3). According to the WOW-security results on **ROPF**, we expect about the most significant half of $\mathbf{y}[i]$ to match the bits of the corresponding **MSDB**-secure ciphertext produced as an intermediate ciphertext.

Next, we simply treat the column $\mathbf{y}$ as ciphertexts emitted by the [7] **MSDB**-secure construction. We carry out all of the pair-wise comparisons, recording when the differing bits are revealed in a vector of guesses $\mathbf{g}$. We can then interpret the vector $\mathbf{g}$ as we did when attacking **MSDB** alone.

RtM. This attack simply reverses the steps, so we sketch the important differences. The outer encryption is now **MSDB**-secure, so one can compute the differing bits on those ciphertexts to get a vector of guesses $\mathbf{g}$ (that will have known and unknown bits recorded). Then we can apply the scale attack to the entries of $\mathbf{g}$ (we leave the unknown bits in place, and bit-shift the string as before). The result will issue guesses for some known bits which we can use as the attack output. We can also replace the unknown bits with 0.5 as in the DistMinAtk to quantitatively approximate the plaintexts.

Selecting a combined scheme. We remark that the constructions achieving **MtR** and **RtM** might not provide equivalent security. (The proof only shows that they achieve *at least* **MSDB** and **ROPF** security separately, but the

combined modes might be strictly stronger.) It was unclear which will be better in practice.

There is, however, an efficiency difference between the existing instantiations of **MtR** and **RtM**. In [7], **MSDB**-secure ORE is constructed with relatively short ciphertexts (about 1.58 times the size of a plaintext), but the OPE version of their scheme has much longer ciphertexts (it expands the plaintext by a factor λ which corresponds to the "security parameter" and may be set to 80 or much larger). Thus **RtM**, which does not require the **MSDB**-secure part to be OPE, results in a much more efficient construction, and it is the one we evaluate below.

Evaluation. In Fig. 9 we plot the performance of the combined attack on **RtM** encryption. The results are similar to Fig. 8 so we conclude that, on this type of data, **MSDB** and **RtM** provide similar security.

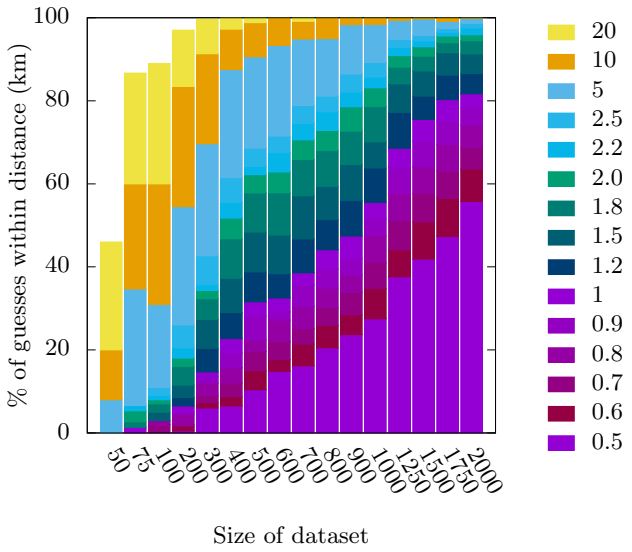


Figure 9: Accuracy of DistMinAtk against **RtM** on `Cal`.

4.6 RtM on timestamp data

We now turn to less disperse data encrypted under **RtM**. We randomly draw time-stamps from the `SpitzTime` dataset, which are distributed over a several-month period. Our attack first encrypts this data using **ROPF** then models the **MSDB** leakage. From this model we then generate our guesses and compare these values to the plaintext data.

The analysis of time-stamp data can not be performed with distances between the guessed and actual values as done for the random locations. This is because the plaintexts are concentrated in a narrow portion of the domain, resulting in all ciphertexts sharing a significant matching prefix. Instead of a direct difference, our analysis and attack are a form of distance windowed one-wayness adversary[5]. Informally, we compare the guessed distance between each pair of ciphertext (in order) with the distance between the plaintext data. More formally, for the sorted vector of guessed values, g, and matching plaintexts, p, the metric of interest is $|(g_i - g_{i-1}) - (p_i - p_{i-1})|$.

We now describe the setup, attack, and results. First, `SpitzTime` data beginning times are parsed into a 32 bit number of seconds starting at an epoch of January 1, 2000. All data is OPE encrypted and shifted as with the scaling attack. Then the **MSDB** leaks are modeled and inferences made. Finally, we generate the guessed data set for comparison.

The results of the attack are shown in Fig. 10. Even for extremely small databases, many time-stamp differences are accurately guessed to within one-hour. The vast majority of guesses were are correct to within two days.

4.7 Modular ORE on real locations

Modular ORE (MORE) was suggested by Boldyreva et al. [5] to address the leakage of **ROPF**-secure ORE. When an adversary sees a column of data encrypted with **ROPF**, it can extract about half of the plaintext via their scaling attack. Thus, they suggest modifying an **ROPF**-secure ORE $\mathcal{E}^{\mathrm{r}}$ as follows: In addition to the usual key K, store another string j, chosen at random from the plaintext space $\{0,1\}^m$. Then define $\mathcal{E}^{\mathrm{mod}}_{(K,j)}(x) := \mathcal{E}^{\mathrm{r}}_K(x+j)$, where the addition $x+j$ is computed modulo 2^m. The construction $\mathcal{E}^{\mathrm{mod}}$ is no longer strictly speaking an ORE scheme, since the addition wraps sometimes. It was shown that efficient range queries are still possible (see [5]). The scaling attack fails completely against $\mathcal{E}^{\mathrm{mod}}$ because it recovers the higher-order bits of $x+j \mod 2^m$, which are independent of x. Recent work [11] developed query protocols to hide the shift value but left the actual encryption algorithm as defined above.

We took the `Cal` dataset, which Mavroforakis et al. used in testing their MORE algorithms, and applied their MORE construction to the latitude and longitude columns independently (with different keys and different "shifts" for each columns). Then we applied the scaling attack to the ciphertexts and plotted the results in Fig. 11. We observe that the fine details of the data are preserved (as expected, because **ROPF**-OPE was used), but it is also obvious how to correct both shifts by hand. We did not explore a quantitative or automated way to remove the shift. The general issue appears to be that for some distributions, shifting by a random value still preserves enough structure so that the shift is easily corrected.

Acknowledgements. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and Space and Naval Warfare Systems Center, Pacific (SSC Pacific) under contract No. N66001-15-C-4070.

This work was supported by NSF grant CNS-1453132.

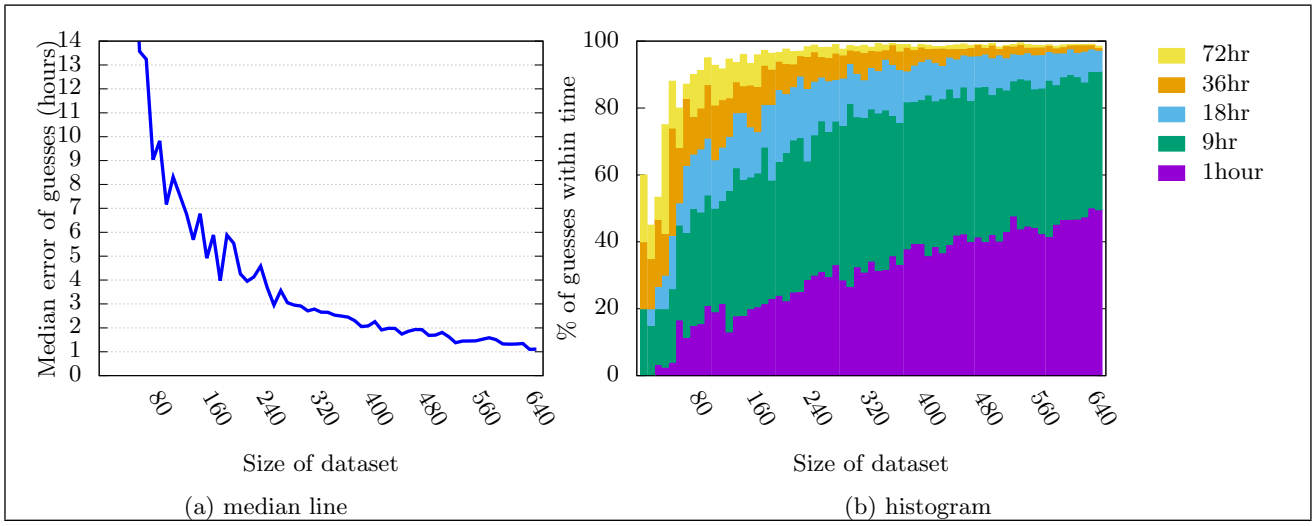
5. REFERENCES

[1] encrypted-bigquery-client. https://github.com/google/encrypted-bigquery-client, 2015.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 563–574. ACM, 2004.

[3] A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal security with cipherbase. In *CIDR*, 2013.

[4] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 224–241. Springer, Heidelberg, Apr. 2009.



(a) median line

(b) histogram

Figure 10: Accuracy of **RtM** leakage on `SpitzTime`.

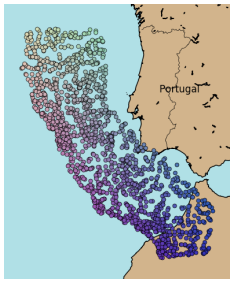


Figure 11: Visualization of MORE on the `Cal` dataset.

[5] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 578–595. Springer, Heidelberg, Aug. 2011.

[6] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 563–594. Springer, Heidelberg, Apr. 2015.

[7] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical order-revealing encryption with limited leakage. In *FSE*, 2016. To appear.

[8] P. Grofig, I. Hang, M. Härterich, F. Kerschbaum, M. Kohler, A. Schaad, A. Schröpfer, and W. Tighzert. Privacy by encrypted databases. In *Privacy Technologies and Policy - Second Annual Privacy Forum, APF 2014, Athens, Greece, May 20-21, 2014. Proceedings*, pages 56–69, 2014.

[9] F. Kerschbaum. Frequency-hiding order-preserving encryption. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 656–667. ACM Press, Oct. 2015.

[10] F. Kerschbaum and A. Schröpfer. Optimal average-complexity ideal-security order-preserving encryption. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 14*, pages 275–286. ACM Press, Nov. 2014.

[11] C. Mavroforakis, N. Chenette, A. O'Neill, G. Kollios, and R. Canetti. Modular order-preserving encryption, revisited. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 763–777, 2015.

[12] M. Naveed, S. Kamara, and C. V. Wright. Inference attacks on property-preserving encrypted databases. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 644–655. ACM Press, Oct. 2015.

[13] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477. IEEE Computer Society Press, May 2013.

[14] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles 2011, SOSP 2011, Cascais, Portugal, October 23-26, 2011*, pages 85–100, 2011.

[15] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000.