Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080501

Securing Databases by using Diagonal-based Order Preserving Symmetric Encryption

Santi Martínez[*], *Josep M. Miret, Rosana Tomàs and Magda Valls*

Departament de Matemàtica, Universitat de Lleida, C. Jaume II, 69, 25001 Lleida, Spain

Received: 4 Aug. 2013, Revised: 1 Nov. 2013, Accepted: 2 Nov. 2013
Published online: 1 Sep. 2014

Abstract: The amount of information stored in databases is constantly increasing. Databases contain multiple records, each of them divided in several data fields. And some of these fields may contain sensitive information, so there is a need to prevent free access to it. Traditionally, cryptography has been used to conceal this kind of information, but conventional cryptography has the problem that, for queries that need access to a specific field for all the records, it requires the decryption of the entire data field. Order preserving encryption ensures that comparing encrypted data returns the same result than comparing the original data. This permits to order encrypted data without the need of decryption. In this way, databases using this kind of cryptosystems admit encrypted record fields while still allowing searches and range queries. In this paper, we propose an order preserving symmetric encryption scheme whose encryption function is recursively constructed. Starting with the trivial order preserving encryption function, which is the identity, a function is constructed in a series of steps by making it more and more complex until the the desired security level is reached. The security of the proposed cryptosystem is also analyzed.

Keywords: databases, privacy, symmetric key, lightweight cryptography, order preserving encryption

1 Introduction

Cryptography allows hiding sensitive information from potential attackers. However, the usage of encrypted information can respond to different needs which require different cryptographic techniques. In particular, this work focuses on database privacy.

Notice that database security is essential to prevent unauthorized access to sensitive information. As an example, a real incident is exposed in The Toronto Star [1], which explains that a bank sold a hard disk on eBay, forgetting to delete plain data stored from hundreds of its customers; the buyer, upon realizing this, tried to resell the disk on eBay.

In secure databases, sometimes one needs to permit certain operations, or queries, that would require the database to decrypt all the fields necessary to perform them, e.g. obtaining records with a field ranging between two values.

Order Preserving Encryption (OPE) allows to perform order comparisons directly on encrypted data, so it ensures that ciphertexts retain the order established between plaintexts. So, if a field is encrypted in this way, SQL range queries [2] can still be done efficiently, while ensuring that an attacker with access to the information stored in the database cannot obtain information about the data in clear.

Consider an encrypted medical database and suppose we want to know how many patients are in an age group. If the cryptosystem used is not order preserving, performing the query requires that we encrypt each of the values belonging to the range of the age group and then compare them with the corresponding fields in the encrypted database, or, alternatively, decrypt the age field for all the records, and then test whether each one is within the range of the query (if the encryption algorithm is not deterministic only the second alternative is valid).

But, if the database uses OPE, we need to encrypt only the first and last values of the range, and test how many records have their encrypted age field within these two encrypted values.

This becomes even more necessary if, instead of working with integer valued fields, such as age, we work with real valued fields. E.g. in the medical database example, we could be interested in the percentage of patients whose blood sugar level is over a certain threshold.

[*] Corresponding author e-mail: santi@matematica.udl.cat

In essence, an OPE scheme is a strictly increasing function that goes from the set to which data belongs to the set to which the ciphertexts belong. Its security relies in that this function, while maintaining order, looks as random as possible [3]. This will ensure that only those with an exact knowledge of how to compute it (which is determined by the cryptosystem key) will be able to invert it.

OPE schemes are necessarily symmetric, since the knowledge of the encryption function permits to approximate, to any precision, the decryption function. Notice that, if an attacker with the capability of encrypting arbitrary values wants to decrypt a particular value, she may perform a dichotomic search for the target value, until a satisfactory approximation is reached.

In this paper, we present the Diagonal-based Order Preserving Encryption scheme (DOPE). The scheme has been designed for environments in which there exists the possibility that an intruder can get access to the encrypted database but cannot encrypt/decrypt arbitrary values. Notice that, even if the access control system of the database manager is responsible of deciding which data can be accessed by each user, in general, direct access to the contents of the database cannot always be prevented, due to security breaches.

From the attackers perspective, knowing that a specific field has been ciphered with order preserving encryption provides an useful source of information if they can get access to stored data. If an attacker knows the corresponding encrypted values for a set of values, she may try to approximate the decryption function. So, if she gets access to new encrypted values, she can compute an approximation of the original values.

E.g., an attacker knows x_1, x_2, y_1, y_2 and the fact that $y_1 = Enc(x_1)$ and $y_2 = Enc(x_2)$; if she obtains an encrypted value y, with $y_1 < y < y_2$, she then knows that its decryption, x, lies somewhere in the interval (x_1, x_2). Moreover, she can interpolate (using, for example, linear interpolation) a value x' whose closeness to x will depend on the predictability of the function and the distance between the values she knew beforehand.

Thus, order preserving encryption functions should minimize this problem ensuring a high level of *unpredictability* by being as random as possible [3].

The rest of the paper is structured as follows: Section 2, provides some related works. In Section 3, the DOPE scheme is presented. Section 4, discusses the security of this cryptosystem. Experimentation results are provided in Section 5. Finally, conclusions are given in Section 6.

2 Related Work

During the last decade, the topic of Order Preserving Encryption has been attracting research interest, with several scientific papers published on this subject.

An initial approach by Bebek is found in [4], where the author proposes a method that allows the encryption of an integer p by adding the p first values of a secure pseudo-random sequence of positive integers. However, the cost of encrypting an n bits value p is exponential in n (since it is linear in p).

Moreover, suppose that μ is the mean of the distribution followed by the pseudo-random sequence (for a uniform distribution on the interval $[1, Max]$, μ will be $\frac{Max+1}{2}$), then the function $f(x) = \mu x$ would approximate the encryption function (and $f^{-1}(x) = x/\mu$ would approximate the decryption function). This approximation will be less useful if μ is close to 0 and the distribution has a large standard deviation.

In [5], Ozsoyoglu et al. propose the use of polynomials for the encryption of integers. These polynomials must have no extremum in the interval at which the data belongs. However, it may be impossible to obtain the formula for the inverse of some polynomials. So, the authors propose to compose various invertible polynomials, so that decryption consists on applying the inverses in reverse order.

To avoid integer overflow errors, they propose controlling the coefficients and using logarithms, which requires dealing with floating point values and precision errors. This makes the election of the key a complex process. Besides, decryption is harder than encryption, since several roots must be computed.

Agrawal et al. [6] propose the transformation of data that follows certain statistical distribution into ciphertexts which maintain the order and follow a different distribution, chosen by the user.

In order to generate the encryption function, they use all the data to be encrypted and a list of samples of the target distribution. Hence, key generation time is linear in the size of the database. When encrypting, data is first transformed in a uniform distribution, which is then transformed into the target distribution. To do this, they split data in several partitions or buckets and use linear interpolation inside them.

If a large amount of data is added to the database, after a key has been generated, it may be necessary to choose a new key and re-encrypt the database.

In [7], Lee et al. propose the COPE scheme (*Chaotic Order Preserving Encryption*). In this scheme, bucket order is randomized according to the key, so, in fact, it is not a pure OPE scheme. The fact that buckets must be sorted to answer a query may affect negatively the cost.

Boldyreva et al. [3] presented an order preserving encryption function relying on the use of a sampling algorithm for the hypergeometric distribution (according to Yum and Lee [8], the sampling algorithm is called less than $\log M + 3$ times on average, where M is the size of the plaintext space).

They point out the fact that, for integer OPE functions, the output set is larger than the input set (which permits that no two clear values correspond to the same encrypted

value). So, a function from $\{1,...,M\}$ to $\{1,...,N\}$, $N > M$, can be uniquely determined by the selection of a subset (of size M) of the output set which contains the encryption of the values of the input set.

More importantly, they proposed a criteria that a good OPE function must fulfill, based on the different ways this selection can be performed (which is related to the negative hypergeometric distribution), noting that, basically, the function must be "as random as possible", while maintaining order.

Martínez et al. proposed two methodologies designed to analyze the quality of an order preserving encryption function [9]. The first one is based on the conversion of the encryption function to a sequence to be analyzed as a noise signal. The second one is based on the differences between the encryption function and the approximations an attacker may compute from a small set of known (correct) points. These differences are called the *unpredictability* of the function. Both techniques consider encryption functions accepting data belonging to the real interval $[0,1]$. But other functions can be mapped accordingly in order to perform the analysis.

An order preserving cryptosystem, whose encryption function derives from the noise analysis technique, was also presented.

In this paper, we propose a more efficient order preserving encryption scheme. In order to evaluate the security of our approach we will quantify its randomness by means of the *unpredictability* metrics.

3 Diagonal Order Preserving Encryption

This section presents our Diagonal-based Order Preserving Encryption scheme (DOPE), including some basic considerations (Section 3.1), the construction of the encryption function (Section 3.2), the encryption and decryption processes (Section 3.3), and the key selection criteria (Section 3.4).

3.1 Basic Considerations

We propose a cryptosystem in which $x_1 < x_2 \Leftrightarrow Enc(x_1) < Enc(x_2)$, where x_1, x_2 are two clear values, so, the order of clear data corresponds to the order of encrypted data. In fact, this is the defining property of a strictly increasing function.

In our scheme, input and output data belong to the real interval $[0,1]$. For this reason, before encryption is done, if input data belongs to a different set it must be mapped into it. Any interval, even the whole $\mathbb{R}$, can be mapped into $[0,1]$ in some way.

Notice that the mapping should be reversed after decryption in order to recover the original data. So, a good mapping function is one that ensures a somewhat uniform distribution over the interval $[0,1]$ (so that entropy is maximized) while being easy to invert.

This mapping process could lead to some information loss (e.g. when the values to be encrypted are character strings), since we are limited by the number of bits of the floating point type used. This problem can be overcome by storing the field in duplicate, using the proposed cryptosystem and an auxiliary one which ensures unambiguous decryption.

Since this mapping is specific to the particular field being encrypted, we will consider this a solved problem, thus focusing in the problem of encrypting values from $[0,1]$ whose distribution is somewhat uniform.

3.2 Key Generation

As has been said, we need a strictly increasing function for encryption. Like all the previous methods, the proposed cryptosystem is also symmetric.

The encryption functions considered in this work are piecewise linear functions, where the number of line segments is related to their security level.

In order to specify one of these functions, we use, as defining points, the list of points at which the function changes direction. This list of points is the key of the cryptosystem.

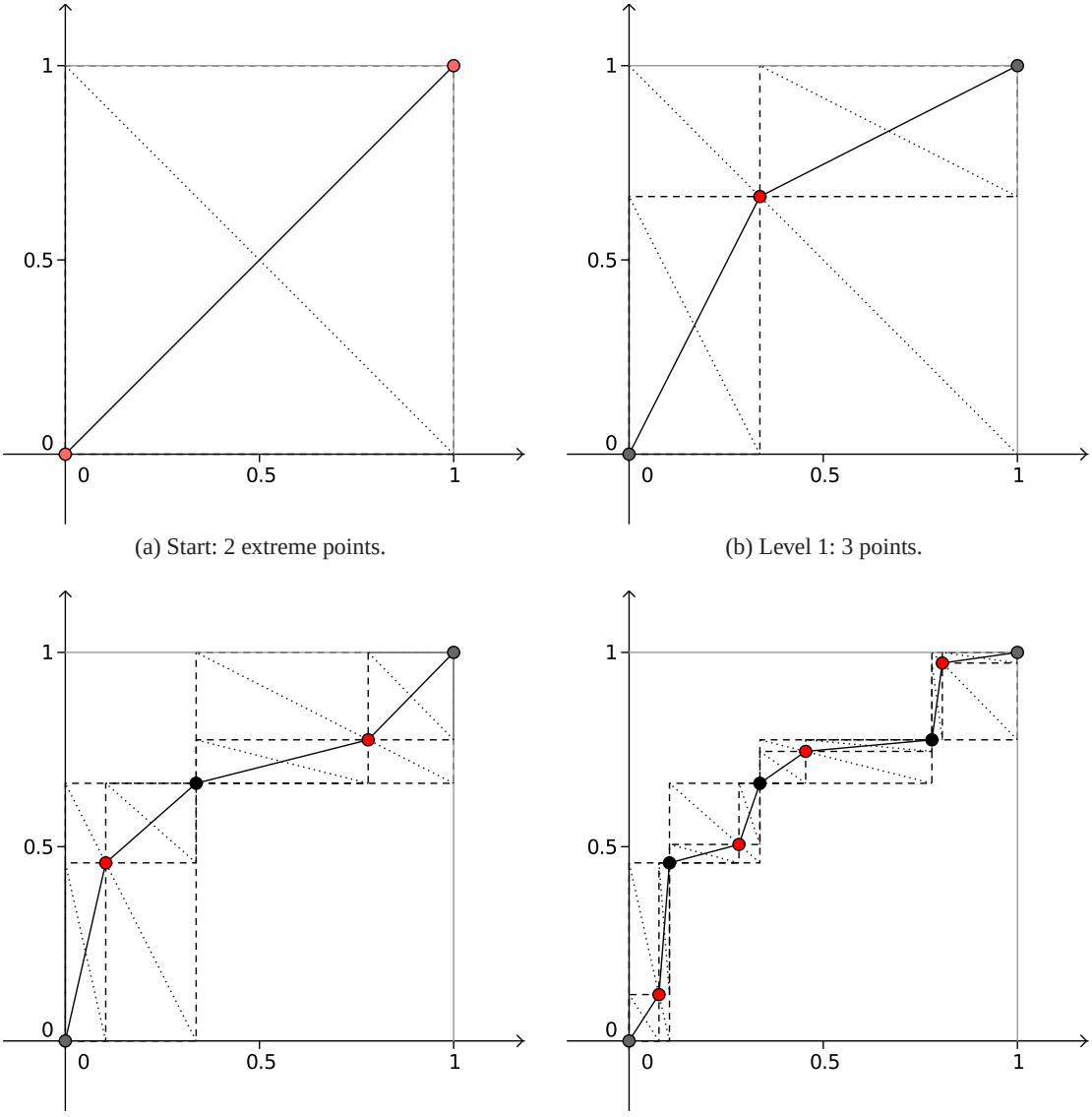
The key generation algorithm proceeds by constructing the encryption function in a series of steps. Each step will add a point to the list. The steps are grouped by levels, so that at each level, the number of points is roughly doubled (see Figure 1).

Initially, the function is the identity defined from $(0,0)$ to $(1,1)$. Notice that the final order preserving function will be enclosed in the square with these two points as opposite vertices (Figure 1a). Moreover, every OPE function going through these two points will cross the descending diagonal, the one from $(0,1)$ to $(1,0)$, exactly once.

So, we randomly choose a point on this diagonal and refine the encryption function, so that it goes through the new point (Figure 1b). This point allows to define two rectangles whose sides are parallel to the axes and with one vertex at the new point and another at one of the extreme points.

Similarly to the initial case, any OPE function going through the current list of points, will be enclosed by the two rectangles, and will cross both descending diagonals once. So, the process can be repeated by refining, at each step, one of the two current function pieces.

Figure 1c shows the result of taking a point on the descending diagonal of each of the two rectangles and refining the encryption function so that it goes through the new points. This corresponds to the second level of the algorithm. The function is now composed by four line segments.



(a) Start: 2 extreme points.

(b) Level 1: 3 points.

(c) Level 2: 5 points.

(d) Level 3: 9 points.

Figure 1: Key generation steps.

The third level refines the function by choosing four new points inside the rectangles corresponding to the current line segments. The result is shown in Figure 1d.

The process will continue until the desired security level l. The obtained encryption function will have 2^l+1 points (or 2^l line segments). Notice that the obtained function will be increasing.

3.3 Encryption and Decryption

The encryption algorithm requires two steps. It takes as input the cryptosystem key (as a list of points sorted by their abscissas) and the value x to be encrypted.

The first step consists on a dichotomic search to locate the position of the clear value in the list of abscissas. If there exists a point whose abscissa coincides with the value, then the encrypted value will be the ordinate of this point. Otherwise, there are two key points, $P_i = (x_i, y_i), P_j = (x_j, y_j)$, whose abscissas precede and succeed the clear value, $x_i < x < x_j$.

Then, the second step consists on a linear interpolation between these two points to obtain a point whose abscissa is the value to encrypt. So, the ordinate is computed with the formula:

$$y = y_i + (x - x_i)\frac{y_j - y_i}{x_j - x_i}. \quad (1)$$

This ordinate is the encryption of x.

The decryption algorithm is almost identical. Its input is the cryptosystem key and the value y to be decrypted.

Note that, although the key points are sorted by their abscissas, using the ordinates yields, obviously, the same order.

So, the algorithm proceeds with the same two steps. First, a dichotomic search to locate, in this case, the position of the encrypted value in the list of ordinates. Second, the interpolation of the preceding and succeeding points to obtain a point whose ordinate is the value to decrypt:

$$x = x_i + (y - y_i)\frac{x_j - x_i}{y_j - y_i}. \tag{2}$$

The corresponding abscissa is the decryption of y.

3.4 Restrictions on Key Selection

In this section, we will show the problems that arise from an unrestricted key generation process and how to solve them. In order to do this, some example functions have been generated. Their keys have 4095 random points plus the two extreme points $(0,0)$ and $(1,1)$. In a real environment, larger keys are recommended, but due to the way key generation works, the difference between keys of 4097 points and larger ones is not noticeable with the naked eye.

Figure 2 shows two of these functions. Each key point belongs to the descending diagonal of a rectangle defined in a lower level (as proposed in Section 3.2). The point selection follows a uniform distribution along the length of the diagonal.

While the function in Figure 2a seems more or less acceptable (although it has some deficiencies), the one in Figure 2b is clearly problematic. It has some flat intervals and jumps, and has the additional problem of being too low during most of its range.

In Figure 3, there are two more extreme examples. We call an encryption function of this kind a *degenerate function*. It can come in two flavors: a function that remains almost flat for most of its domain, approaching the point $(1,0)$, and then grows abruptly to the point $(1,1)$, as can be seen in Figure 3a, or a function that grows fast, approaching the point $(0,1)$, and then remains almost flat until reaching the point $(1,1)$, similar to Figure 3b. Degenerate functions are less random, and, because of this, less useful.

Moreover, taking such functions for the encryption process could be a problem since they may carry a greater information loss in the least significant bits. Notice that, in jumps, there is a risk of two consecutive key points having the same abscissa (since floating point numbers have a limited number of bits). In the same way, flat intervals may involve consecutive key points with equal ordinates. These two facts would hinder encryption and decryption since the interpolation would require a division by zero.

However, if no restriction is applied to the selection of the random points that compose the key, we obtain these functions quite often. So we must modify the key generation process in order to avoid jumps, flat intervals and degenerate functions in general.

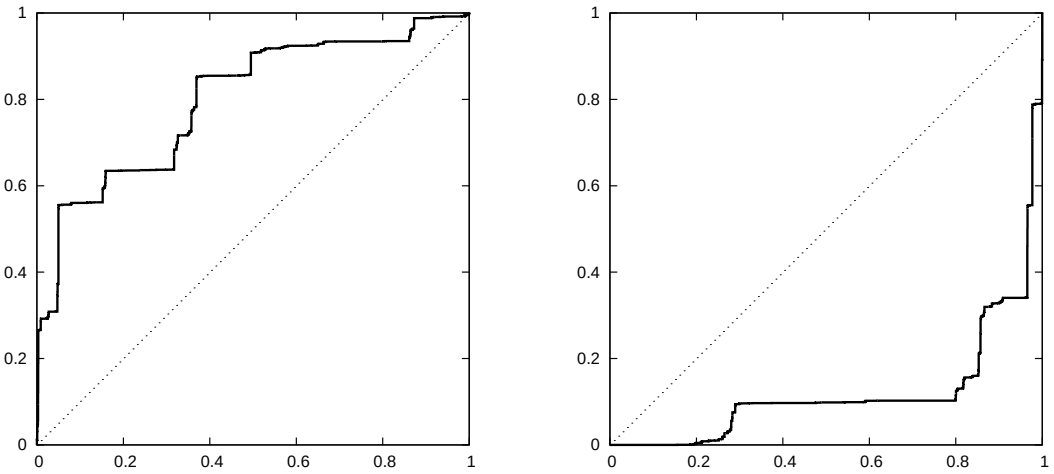
Flat intervals and jumps are caused by the selection of points that belong to the borders of the rectangles defined in the previous level, i.e. choosing a point near the extremes of the diagonal. Notice that, if the new random point is one of the extremes, the new rectangles defined by this point and the previous ones are flat (one vertical and one horizontal). So, even if there are more points to be selected inside them, the shape will remain unaltered.

In order to prevent this, the rectangles will leave a gap on each side to ensure that the points of the higher levels will have enough space. This gap is decreased at each level and does not affect the level 0 points, so, the two first points will still be $(0,0)$ and $(1,1)$. The third point to be generated will be somewhere in the diagonal of a rectangle from (g_1,g_1) to $(1-g_1,1-g_1)$, with g_1 being the initial (level 1) gap. If the selected point is on the upper-left extreme of the diagonal, $(g_1,1-g_1)$, then we must place the next point in a rectangle from (g_2,g_2) to $(g_1-g_2,1-g_1-g_2)$. This rectangle has a width of g_1-2g_2, so, in order to avoid flat rectangles g_2 must be less than $g_1/2$. We propose to use $g_{n+1}=g_n/g_d$, where g_d is a constant that controls the shrinking of the gaps. So, we need to find the appropriate value for the initial gap, g_1, and the gap shrinking factor, g_d. Figure 4 shows this situation.

We will now consider the evolution of the thinner permitted rectangles obtained when the selected points are on the extremes of the diagonals. The level 1 permitted rectangle (which happens to be a square) has a width and height of $1-2g_1$, so, $0 \leq g_1 < 1/2$. Then, the level 2 rectangles will have a width and height of, at least, $g_1-2g_2 = (g_d-2)g_1/g_d$, so, $g_d > 2$. The level 3 rectangles have a minimum width and height of $g_2-2g_3 = (g_1-2g_2)/g_d = (g_d-2)g_1/g_d^2$. And, in general, the minimum width and height of a level $l>1$ permitted rectangle will be $g_{l-1}-2g_l = (g_d-2)g_1/g_d^{l-1}$.

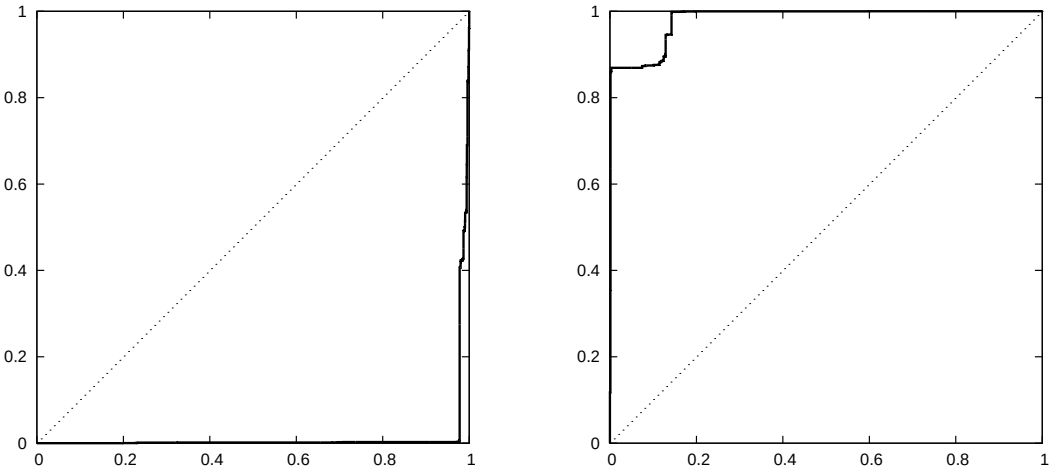
The problem is that it is impossible to maximize all these rectangles at the same time, since, in order to maximize the rectangles of a level l, we need to leave small gaps in this level, which then, if the point is chosen on the extreme, makes the level $l+1$ rectangles smaller. We suggest to use the values $g_1 = 2^{-7}$ and $g_d = 4$. Section 4.2 will justify the choice of these values.

This restriction prevents flat intervals and jumps in the encryption function, so this variation will minimize the appearance of degenerate cases while ensuring a random appearance.



(a) (b)

Figure 2: Examples of unrestricted encryption functions with a 4097 points key.



(a) (b)

Figure 3: Examples of degenerate encryption functions with a 4097 points key.

4 Security of the Encryption Function

OPE functions are strictly increasing functions, so anyone who knows some of its points (i.e. plaintext-ciphertext pairs) can approximate them.

So, better OPE schemes are those in which these approximations are further from the correct function. In [9], we provided a method to evaluate OPE functions following this criterion.

4.1 Attack Scenario

We consider the following scenario: an attacker has gained access to the database and knows all the encrypted values. Besides, she knows the corresponding clear values for some of the encrypted values. Notice that, she knows, at least, that the function goes through the points $(0,0)$ and $(1,1)$.

With these premises, the attacker may interpolate the decryption function between the points she knows, and use it to obtain an approximation of the clear values corresponding to the rest of the database values.

In order to evaluate the security of the proposed cryptosystem, we will use the unpredictability metric [9] to test how good is an approximation of the function when the attacker knows some of the points it goes through.

This metric is based on the continuous mean absolute error (MAE) between the decryption function and its

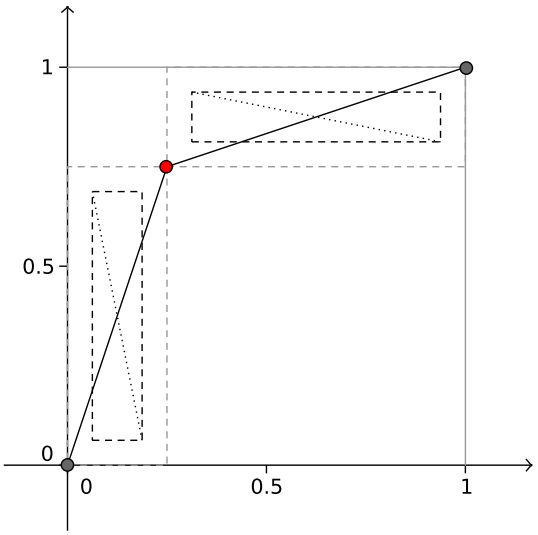


Figure 4: Gaps to prevent equality of consecutive abscissas and ordinates. In this graphic $g_1 = 1/4$, $g_d = 4$.

approximation (so a larger value implies less predictive power). It corresponds to the average of the difference between the decryption function and the approximation over all the function domain.

In order to simplify the analysis, the points known to the attacker (which we call the attacked points) are considered with equally spaced ordinates. Thus, for any quantity of attacked points a specific unpredictability may be computed. Then, we can evaluate the general unpredictability of the encryption function by computing the specific unpredictabilities for different sets of attacked points.

In the worst case, when the attacker can approximate the function very well, the MAE will tend to 0. In general, for an attack with $n+1$ known points with equally spaced ordinates, the maximum MAE is $\frac{1}{2n}$. It is achieved when the decryption function is a staircase whose only jumps are at the attacked points. However, functions of this kind are not adequate, since they collapse most of the clear values to a small set of possible encrypted values.

So, the best functions are those whose specific unpredictabilities for most sets of attacked points correspond to MAEs that lie somewhere in between the minimum (0) and the maximum ($\frac{1}{2n}$). In [9], we proposed that the recommended MAEs should be the average of the two values: $\frac{1}{4n}$.

4.2 Security Analysis Results

We have analyzed the unpredictability of the DOPE scheme, using different key lengths and combination of parameters (g_1 and g_d). For each one, we have created five different keys, and, for each key, we have considered several approximations of the decryption function with different amounts of attacked points. The number of attacked points considered has been 2, 3, 4, 6, 9, 13, 19, 28, 42, 63, 94, 141, 211, 316, 474, 711, 1066 and 1599 (each value is 1.5 times the previous value, rounding down).

With these experiments we have found the best combination of parameters in order to achieve more unpredictable encryption functions. The experiments performed show that, as stated in Section 3.4, the best initial gap and gap shrinking factor are $g_1 = 2^{-7}$ and $g_d = 4$. From now on, when we refer to DOPE we will assume this configuration.

Figures 5, 6 and 7 show the averaged MAEs that have been obtained. Each line (except the two higher ones) corresponds to a particular key length. The horizontal axis corresponds to the number of attacked points and the vertical axis corresponds to the average of the MAE between five pairs of decryption function and approximation. Both axis are in logarithmic scale.

The two higher lines are the maximum and recommended MAE. So, the higher the lines, the more unpredictable are the functions. But, if a function approached the maximum MAE it would mean that it is a staircase.

In general, the larger the key, the less predictable is the function. This is more noticeable when the attacker knows a bigger amount of points.

Figure 5 compares DOPE with the unrestricted gapless DOPE (with $g_1 = 0$). The fact that the restricted version avoids degenerate functions is what causes its better results. Notice that, with a null initial gap, any subsequent gap will also be zero regardless of the value of g_d.

Although, we have experimented with several combinations of parameters, here we will only show the graphics of those that were closer to the chosen configuration.

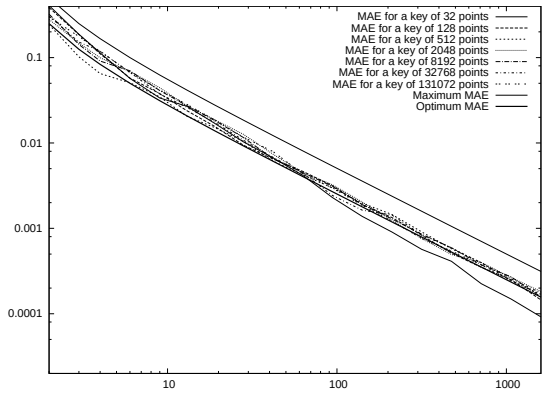
Figure 6 shows the unpredictability of DOPE with smaller and larger initial gaps, but with the same gap shrinking factor. As can be seen, both configurations give results that are worse than standard DOPE.

Figure 7 shows the unpredictability of DOPE with smaller and larger gap shrinking factors, but with the same initial gap. In this case, the results obtained are also worse than those of the standard version of DOPE.

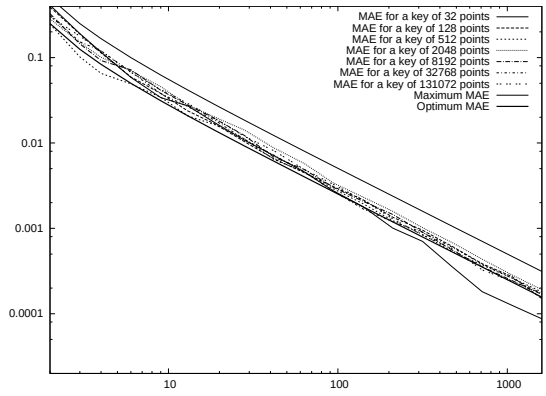
This means that the corrections performed in order to avoid degenerate functions and the selection of parameters are fairly accurate.

5 Experimentation

In order to test the performance of the presented cryptosystem, we have conducted a set of experiments. We present the experimentation results obtained from the implementation of the proposed DOPE scheme. Different amounts of points have been used to test the efficiency of

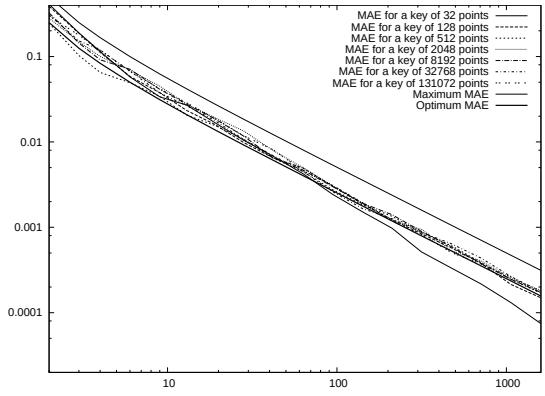


(a) $g_1 = 2^{-7}$, $g_d = 4$.

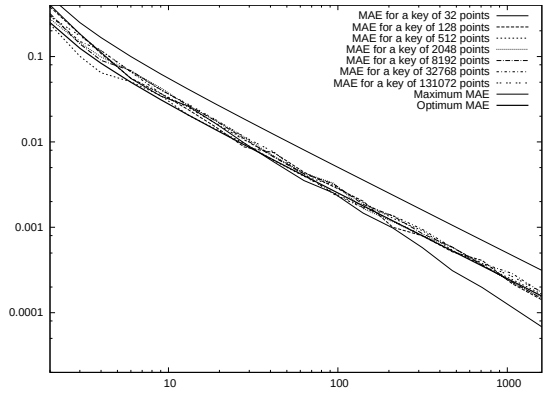


(b) $g_1 = 0$.

Figure 5: Unpredictability of DOPE and gapless DOPE.



(a) $g_1 = 2^{-8}$, $g_d = 4$.



(b) $g_1 = 2^{-6}$, $g_d = 4$.

Figure 6: Unpredictability of DOPE with different initial gaps.

key generation and encryption/decryption, when key length increases.

Table 1 shows the results for numbers of points of the form 2^l, for $4 \leq l \leq 27$, using a 2.4 GHz computer. It reflects the number of points of the key, its resulting size, the key generation time, $GenT$, and the encryption/decryption time, $EncT$. The values to encrypt and the key points coordinates have been represented as 64 bit floating point numbers (type `double` in the C language), so, the size of a key, in bytes, is 16 times its number of points.

In order to obtain the key generation time, we generated ten encryption functions for each value of l and computed the mean of their generation times.

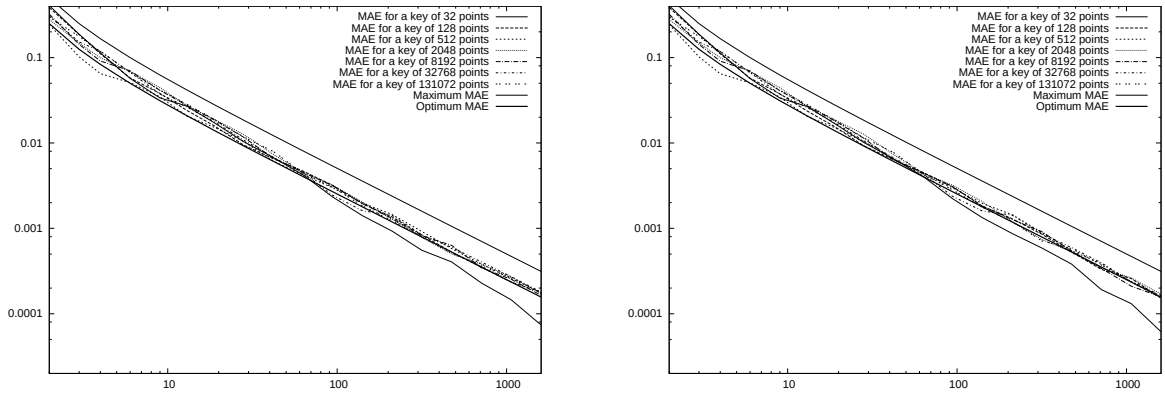
To measure the encryption time of a single value, each of the functions has been used to encrypt a large set of one million values between 0 and 1. Then, we computed the mean of the encryption times of these sets, divided by the number of values they contain (i.e. 1000000), so that it corresponds to the encryption time of a single value.

Key generation times are linear in the number of points, especially for larger keys. For very small ones there are some overheads associated which increment their generation times.

As expected, encryption times are logarithmic in the number of points. And decryption times (not shown) were almost identical. Even encrypting a large amount of data at once, the obtained times are acceptable. E.g. with a key of 65537 points ($l = 16$), the time needed to encrypt one million values would be of about 62.9 ms ($1000000 \cdot 62.9$ ns). So, the cryptosystem is considerably fast.

Key generation is much slower than the encryption of a single value, but it only takes place occasionally, so its cost is affordable. A key of 65537 points will take 3.85 ms to generate, and will occupy 1 MiB.

If key size were a problem, the seed used in the pseudo-random generator can be stored in its place. This way, it can be generated again whenever needed (maintaining it in memory the maximum time possible, in order to reduce the number of regenerations). Obviously, the seed (or the key) must be stored in a secure way.



(a) $g_1 = 2^{-7}$, $g_d = 5$. (b) $g_1 = 2^{-7}$, $g_d = 3$.

Figure 7: Unpredictability of DOPE with different gap shrinking factors.

Table 1: DOPE Experimentation Results

l	Num. Points	Key Size	GenT	EncT
4	17	272 B	18.1 μs	15.39 ns
5	33	528 B	19.1 μs	18.89 ns
6	65	1.02 KiB	21.0 μs	22.05 ns
7	129	2.02 KiB	23.9 μs	25.78 ns
8	257	4.02 KiB	31.4 μs	29.39 ns
9	513	8.02 KiB	47.6 μs	34.72 ns
10	1025	16 KiB	74.5 μs	39.08 ns
11	2049	32 KiB	131.1 μs	43.27 ns
12	4097	64 KiB	247.6 μs	47.58 ns
13	8193	128 KiB	490.4 μs	51.30 ns
14	16385	256 KiB	976.9 μs	55.09 ns
15	32769	512 KiB	1.922 ms	59.30 ns
16	65537	1 MiB	3.853 ms	62.92 ns
17	131073	2 MiB	6.954 ms	66.76 ns
18	262145	4 MiB	15.92 ms	71.25 ns
19	524289	8 MiB	32.65 ms	77.22 ns
20	1048577	16 MiB	64.84 ms	84.77 ns
21	2097153	32 MiB	129.5 ms	91.89 ns
22	4194305	64 MiB	258.7 ms	99.28 ns
23	8388609	128 MiB	515.5 ms	108.1 ns
24	16777217	256 MiB	1.026 s	121.0 ns
25	33554433	512 MiB	2.050 s	149.9 ns
26	67108865	1 GiB	4.091 s	194.6 ns
27	134217729	2 GiB	8.166 s	220.7 ns

6 Conclusions

In this paper, a new order preserving encryption (OPE) scheme has been proposed. It encrypts data belonging to the real interval $[0,1]$, so, if input data belongs to a different set, it must be mapped into it. With this cryptosystem, encryption and decryption times are logarithmic in the number of key points, which has been demonstrated by means of an experimentation.

A security analysis has been performed in order to test the unpredictability of the obtained functions and select the best parameter configuration for the restrictions in key generation. This allowed to avoid degenerate cases, so that the desired entropy can be obtained.

An attacker with access to the contents of a database with some fields using OPE can correctly order the records by these fields (which is unavoidable, as this is the defining property of OPE). In order to avoid the exposition of sensitive information, an important recommendation is to encrypt all the fields, using one cryptosystem or another depending on the type of queries that should be allowed.

Thus, if a field is not searchable, it should be encrypted using a non-deterministic cryptosystem, so that same data produce different ciphertexts. E.g.: a database of people may contain a 'hair_color' field, but disallow searches of the type "hair_color=chestnut".

For fields in which searches by equality are allowed but interval searches are not permitted (or make no sense), a deterministic cryptosystem must be applied. E.g.: 'city_of_birth' could be encrypted in this way.

Fields in which interval searches are allowed should be ciphered with an OPE scheme, like the proposed in this paper. E.g.: 'year_of_birth' should be encrypted in this way, in order to be able to search by filtering age ranges.

As said in Section 3.1, both in the conversion of the data that needs to be encrypted into a floating point value, and in the encryption process, the least significant bits can be modified, introducing small errors. In most situations, this will not be a problem, e.g. for the field 'year_of_birth' belonging to the integer interval $[1890, 2145]$, only the first 8 bits will be relevant, so that small errors will not affect at all. However, with fields requiring more precision, some information could be lost.

In situation in which these small errors can be a problem, it is a good practice to store the field in duplicate, using the proposed cryptosystem and an auxiliary one which ensures unambiguous decryption.

Finally, if the database contains multiple sortable fields, it is recommended to encrypt each of them with a distinct key, to hinder the approximation of the encryption function by attackers with access to plaintext-ciphertext pairs of different fields.

Acknowledgement

This work was partly supported by the Spanish Government through project MTM2010-21580-C02-01, and the Government of Catalonia under grant SGR2009-442.

References

[1] Tyler Hamilton. Error sends bank files to eBay. The Toronto Star, September 15, 2003.

[2] James Groff and Paul Weinberg. *SQL The Complete Reference, 3rd Edition*. McGraw-Hill, Inc., 2010.

[3] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. Order-Preserving Symmetric Encryption. *Advances in Cryptology - EUROCRYPT 2009*, LNCS(5479):224–241, 2009. ISSN 0302-9743.

[4] Gürkan Bebek. Anti-tamper database research: Inference control techniques. Technical Report EECS 433, Case Western Reserve University, 2002. Final Report.

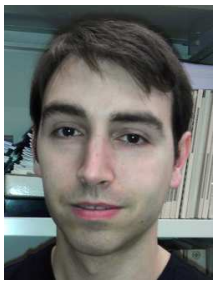
[5] Gultekin Ozsoyoglu, David A. Singer, and Sun S. Chung. Anti-Tamper Databases: Querying Encrypted Databases. In *17th Annual IFIP WG 11.3 Working Conference on Database and Applications Security*, 2003.

[6] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order Preserving Encryption for Numeric Data. In *ACM SIGMOD international conference on Management of data*, pages 563–574, 2004.

[7] Seungmin Lee, Tae-Jun Park, Donghyeok Lee, Taekyong Nam, and Sehun Kim. Chaotic Order Preserving Encryption for Efficient and Secure Queries on Databases. *IEICE Transactions on Information and Systems*, E92-D(11):2207–2217, 2009.

[8] Dae Hyun Yum and Pil Joong Lee. On the average cost of order-preserving encryption based on hypergeometric distribution. *Information Processing Letters*, 111(19):956–959, July 2011.

[9] Santi Martínez, Josep M. Miret, Rosana Tomàs, and Magda Valls. Security Analysis of Order Preserving Symmetric Cryptography. *Applied Mathematics & Information Sciences (AMIS)*, 7(4):1285–1295, July 2013. ISSN 1935-0090.



Santi Martínez is a Postdoctoral Research Assistant at Universitat de Lleida. He received his B.Sc. and M.Sc. in Computer Science from Universitat Rovira i Virgili in 2002 and 2004, respectively, and his Ph.D. in Engineering from Universitat de Lleida in 2011. His research interests include cryptography, RFID systems and elliptic curve cryptosystems.

Josep M. Miret is an Associate Professor at Universitat de Lleida. He received his M.Sc. in Mathematics from Universitat de Barcelona in 1983, and his Ph.D. in Mathematics from Universitat Politècnica de Catalunya in 1999. Since 1990, he is at Universitat de Lleida where he is leading the Cryptography and Graphs Research Group. He has organized several conferences on applied mathematics and cryptography. His research interests include information security, cryptography with elliptic and hyperelliptic curves and its computational aspects.

Rosana Tomàs is an Assistant Professor at Universitat de Lleida. She received her B.Sc. in Computer Science from Universitat de Lleida in 2002, her M.Sc. in Computer Science from Universitat Rovira i Virgili in 2004, and her Ph.D. in Engineering from Universitat de Lleida in 2011. Her research interests include cryptography, smart cards security and elliptic curve cryptosystems.

Magda Valls is an Associate Professor at Universitat de Lleida. She received her B.Sc. in Mathematics from Universitat Autònoma de Barcelona in 1994, and her M.Sc. and Ph.D. in Applied Mathematics from Universitat Politècnica de Catalunya in 1999 and 2001, respectively. Her research interests include cryptography, computational security and elliptic curve cryptosystems.