Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070403

Security Analysis of Order Preserving Symmetric Cryptography

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

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

Received: 26 Feb. 2012, Revised: 4 Mar. 2013, Accepted: 17 Mar. 2013
Published online: 1 Jul. 2013

Abstract: Order preserving encryption is a type of homomorphic encryption in which the homomorphic operation is order comparison. This means that comparing encrypted data returns the same result than comparing the original data. This allows to order encrypted data without the need of decryption. A possible use for this kind of cryptosystems is in databases, where a record field may be encrypted and still permit range queries. An important problem is determining how good a particular order preserving encryption scheme is. In fact, characteristics of order preserving cryptosystems make traditional security analysis useless. In this paper, we propose two different methodologies, applicable to most order preserving schemes, that can be used to determine their security by analyzing their randomness. The first one relies on techniques of noise analysis by converting the encryption function into a noise signal whose power distribution will be considered. The second one relies on techniques of error analysis. It is based on the computation of the mean absolute errors between the encryption function and several approximations defined by small sets of plaintext-ciphertext pairs. As a derived result of the first of these methodologies, a new order preserving cryptosystem is proposed.

Keywords: databases, privacy, lightweight cryptography, order preserving, security analysis

1 Introduction

Nowadays, the amount of information stored in databases is constantly increasing. In a database, each record has several data fields, some of which may contain sensitive information, so there is a need to prevent free access to it.

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. Frequently, one needs to permit these kind of operations, e.g. obtaining records with a field ranging between two values.

Homomorphic cryptosystems allow to perform one (or sometimes more) operation without decrypting the operators. The permitted operation depends on the particular cryptosystem.

Order Preserving Encryption (OPE) allows to perform order comparisons, so it ensures that ciphertexts retain the order established between plaintexts. So, if a field is encrypted in this way, SQL range queries [1] can still be performed efficiently, ensuring that an attacker with access to the information stored in the database cannot obtain information about the data in clear.

In essence, an order preserving cryptosystem is a strictly increasing function that goes from the set to which data belongs to the set to which the ciphertexts belong. The security of the cryptosystem relies in that this function, while maintaining order, looks as random as possible [2]. 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.

All order preserving cryptosystems are necessarily symmetric, since the knowledge of the encryption function would permit 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 provide two methodologies that are designed to test how random and, hence, secure, an order preserving encryption scheme is. We will take benefit of concepts of noise analysis, in particular the *color* of the noise (i.e. the power distribution in the frequency spectrum) and error analysis, by means of mean absolute errors. As an additional result, we present a new order

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

preserving cryptosystem derived from the first of these two methodologies.

The rest of the paper is structured as follows: In Section 2, it is shown the motivation and a practical example of this kind of cryptography. Section 3 provides some examples of previous cryptosystems of this kind. Section 4 displays some previous mathematical concepts used in this paper. The two methodologies are presented in Sections 5 and 6 (with the new cryptosystem presented in Section 5.1). Section 7 tests the security and efficiency of the new cryptosystem, by means of the proposed methodologies and an experimentation. And, finally, a concluding Section 8.

2 Motivation

Order preserving encryption schemes have been designed for environments in which there exists the possibility that an intruder can get access to the encrypted database, but neither has information on the distribution of clear values nor can encrypt/decrypt arbitrary values. Notice that, even if the access control system of the database manager is responsible of deciding what 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.

A real incident is exposed in The Toronto Star [3], which explains that a bank sold a hard disk on eBay, forgetting to delete data stored from hundreds of its customers. The buyer, upon realizing this, tried to resell the disk on eBay.

Consider a medical database, encrypted (as it should be), 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 of the database, 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 is encrypted with order preserving encryption, we need to encrypt only the first and the last values of the range, and make a query of 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 worked 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.

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.

On the one hand, they can correctly order the data by that field (which is unavoidable, as this is the defining property of OPE). So, in order to avoid the exposition of sensitive information (since even an unusual combination of non-sensitive fields could be used as an identifier), the rest of the fields should also be encrypted, using one cryptosystem or another depending on the type of queries that should be permitted.

On the other hand, 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 some 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 $y_1 = encrypt(x_1)$ and $y_2 = encrypt(x_2)$; if she obtains an encrypted value y, with $y_1 < y < y_2$, she automatically 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 (i.e., its lack of randomness) and the distance between the values she knew beforehand.

So, methods of ensuring the obtaining of good (unpredictable) order preserving encryption functions are urgently needed.

3 Related Work

Order preserving encryption has been attracting interest during this last decade, with several scientific papers published on the subject.

An initial approach 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, this encryption method is inefficient for large numbers and may be predictable in some cases.

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. Moreover, the cost of encrypting a n bits value p is exponential in n (since it is linear in p).

In [5], it is mentioned the possibility of using polynomials for the encryption of integer data. These polynomials must have no maximum or minimum in the interval at which the data to be encrypted belongs. However, it may be impossible to obtain the formula corresponding to the inverse of the polynomial, which would hinder decryption. Therefore, as an alternative, the authors propose to apply, in a specific order, various simple polynomials (of the form $f(x) = ax^b + c$), all of

them invertible (the inverse would be $f(x) = \sqrt[b]{\frac{x-c}{a}}$), so that decryption consists on applying the inverses of these polynomials in reverse order. The authors suggest restricting the maximum degree of the polynomials to 2 and the range of the coefficients to $\{1..32\}$.

Since integers are implemented with a fixed length (unbounded integers are less efficient), integer overflow errors must be avoided. To avoid these errors, the authors propose controlling the parameters that define the polynomials and using logarithmic functions ($f(x) = \log_2 x + c$), which would require dealing with real (floating point) values. This implies that precision errors must also be considered.

In short, the election of the key is a complex process that requires the careful consideration of what parameters can be chosen in order to avoid integer overflow errors and minimize precision errors. Decryption is harder than encryption, since several roots must be computed.

In 2004, R. Agraval et al. [6] propose the encryption of data belonging to a subset $[p_{min}, p_{max})$ of the integers, although they suggest the possibility of treating floating point values as if they were integers (i.e. interpreting a 32 bits `float` as if it were an `int` of the same size), since positives maintain the same order, and for the negatives, that have the order reversed, one should only need to subtract the resulting integer from the largest negative.

The method they propose, transforms data that follows certain statistical distribution into ciphertexts that maintain order and also follow a different distribution, chosen by the user. In order to generate the encryption function, they use all the data to be encrypted (because of that, if the database is initially empty, the administrator has to provide samples of possible expected values), and a list of samples of the distribution that has to be emulated. The auxiliary information necessary for the encryption and decryption of data (i.e. the cryptosystem key) will be generated from all these samples. Moreover, in order to model the distributions, data need to be partitioned in buckets. Inside them, linear interpolation will be used. When encrypting, data is first transformed in a uniform distribution, that is then transformed into the target distribution.

One of the drawbacks of the cryptosystem they propose is key generation. While it is relatively small (its size is three times the number of buckets, and it is assured that no more than 200 are needed), its generation is linear in the size of the database. Moreover, if after a key has been generated a large amount of data is added to the database (which is expected to happen if the key has been created for an empty database), it may be necessary to choose a new key and re-encrypt the database.

In 2009, [7] proposes 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 order preserving encryption. The fact that buckets must be sorted in order to perform queries may affect negatively the cost.

A. Boldyreva et al. [2] presented an order preserving encryption function relying on the use of a block cipher.

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 $[1,N]$ can be 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 needs to be "as random as possible" while maintaining order.

In this paper, we propose two methodologies designed to analyze the quality of an order preserving encryption function. 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 difference between the encryption function and the approximations an attacker may compute from a small set o known (correct) points. A new order preserving cryptosystem is also presented. Its encryption function derives from the noise analysis technique.

4 Basic Considerations

Homomorphic encryption permits that a specific operation between plaintexts corresponds to another operation (the same, in some cryptosystems) between ciphertexts, that is,

$$\begin{aligned} enc(x_1 * x_2) &= enc(x_1) \star enc(x_2), \\ x_1 * x_2 &= dec(enc(x_1) \star enc(x_2)), \end{aligned} \tag{1}$$

where x_1, x_2 are two plaintexts.

In the present case, we focus in cryptosystems in which the homomorphic operation is order comparison, that is,

$$x_1 < x_2 \equiv enc(x_1) < enc(x_2), \tag{2}$$

so, the order of clear data corresponds to the order of encrypted data.

Notice that, since relational operators return a boolean value, there is no need for decryption after applying the operator. Also, any other relational operator will be preserved, since all of them can be defined in terms of $<$ and logical operations, e.g. $(a = b) \equiv \neg(a < b \vee b < a)$.

As said in previous sections, the advantage of order preserving encryption is the fact that if a database needs to be sorted by a field, the order of records will be the same whether the field is encrypted or not.

In fact, property 2 corresponds to the defining property of a strictly increasing function. So, it is needed that the encryption function is strictly increasing.

Monotonically (but not strictly) increasing functions, for which the more relaxed property

$x_1 \leq x_2 \equiv f(x_1) \leq f(x_2)$ holds, are undesirable for cryptography, since they allow that for values $x_1 < x_2$, the result of the function coincides $f(x_1) = f(x_2)$, so, the function is non-injective, and, hence, not invertible. This would rule out the possibility of decryption.

In the analysis presented in this paper, we consider only functions with inputs and outputs belonging to the real interval $[0,1]$. For encryption functions defined outside this interval, their domain and/or codomain should be mapped to this interval before performing the analysis. Any interval, even the whole $\mathbb{R}$, can be mapped in this way.

A good mapping function for the input values is one that ensures a somewhat uniform distribution over the interval $[0,1]$. The knowledge (or lack thereof) of the way in which data will be distributed may encourage one particular mapping function.

Since this mapping is specific to the particular field being encrypted, we will consider this a solved problem for the rest of the paper. So, for the security analysis, we consider encryption functions from $[0,1]$ to $[0,1]$ whose input is uniformly distributed.

Not all the strictly increasing functions would be good candidates for an encryption function. Consider, as an example, the function $y = +\sqrt{x}$, restricted to the interval $(0,1)$. It is strictly increasing, so it is invertible (decryption can be performed) and order preserving, but it may be trivially deduced by an attacker who knew a few points. Good encryption functions need to look random.

So, we need methods to analyze the randomness of order preserving encryption functions and to quantify how much predictable a function is (bearing in mind that such a function should be increasing).

5 Randomness of an Encryption Function

In general, better random generators produce samples that, when considered as a signal correspond to uniformly distributed white noise [8]. So, one of the necessary (but not sufficient) conditions that a random sequence must satisfy in order to be considered truly random is to have a flat spectrum.

But the encryption function of an order preserving cryptosystem cannot be truly random, since it must be strictly increasing, so in order to analyze the encryption function it must be transformed in a way that eliminates this restriction. Moreover, it is not clear that a flat spectrum signal is adequate for this process.

In this section, we propose a transformation of an order preserving encryption function that allows its analysis as a noise sequence and may determine which power distribution (i.e. which colors of noise) work better.

Notice that, even though the function is defined only between $(0,0)$ and $(1,1)$, we can extend it to the complete real plane by considering that the same growth pattern repeats in every other square from (k,k) to $(k+1,k+1)$, for integer k. So, the way the function oscillates around the identity function can be considered a periodic signal and be analyzed as such.

Then, if we rotate the function 45° clockwise, we obtain a function with period $\sqrt{2}$ whose slope takes values between -1 (corresponding to an horizontal slope in the original function) and $+1$ (corresponding to a vertical slope). This slope is the signal that we will analyze.

So, the transformation process of an encryption function consists of the following steps:

1. The function is sampled, and the points (x,y) are rotated 45° clockwise, using the following equations:

$$\begin{aligned} x_r &= (y+x)\cdot\sqrt{2}/2, \\ y_r &= (y-x)\cdot\sqrt{2}/2. \end{aligned} \tag{3}$$

2. The derivative of the rotated function is taken.
3. This final function is sampled again, so that its distribution of frequencies can be evaluated.

After that, a fast Fourier transform (FFT) may be used in order to study the noise pattern of the obtained sequence.

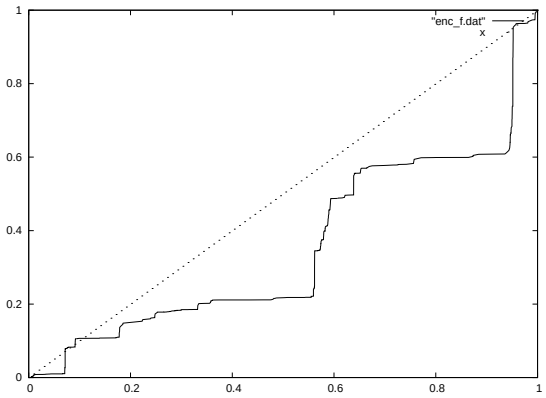


Figure 1: Example encryption function.

As an example, Figure 1 shows an encryption function, whose transformation can be seen in Figures 2 and 3. Notice that, since the original function is defined inside the square from $(0,0)$ to $(1,1)$, the rotated function (Figure 2) is defined inside a square with vertices $(0,0)$, $(\sqrt{2}/2,-\sqrt{2}/2)$, $(\sqrt{2},0)$ and $(\sqrt{2}/2,\sqrt{2}/2)$. In Figure 3, it is shown the derivative (i.e. the slope) of the rotated function.

Then, after sampling the derivative, the FFT (Figure 4) will show how the frequencies are distributed in the spectrum.

In this case, it is clear that the power decreases when the frequency increases (at a rate of almost 6 dB per octave). Noise with this particular spectral density is called red (or Brownian) noise.

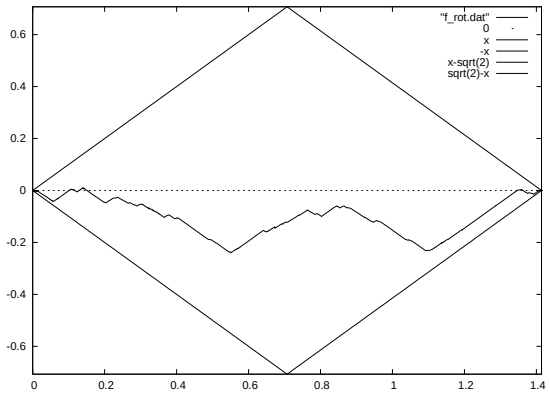


Figure 2: Rotated function.

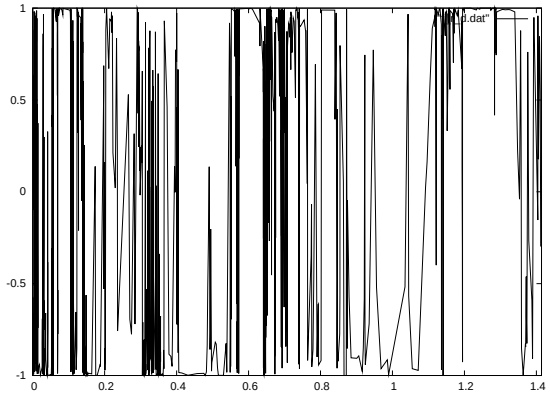


Figure 3: Derivative of the rotated function.

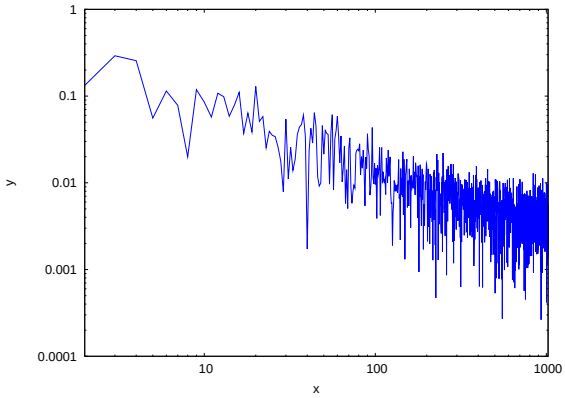


Figure 4: Fourier transform.

5.1 Noise-based Order Preserving Encryption (NOPE)

This transformation process may be reversed (using some necessary modifications), so that, from a particular noise signal, an encryption function can be obtained with the desired characteristics. This has been done, mainly, in order to test which colors of noise produce the best encryption functions.

The generation of a noise based encryption function consists of the following steps:

1. Generate a noise sequence with specific characteristics (e.g. uniform white noise).
2. If necessary, adjust the sequence to the interval $(-1,1)$.
3. The sequence is integrated, using as the x range the interval $[0,\sqrt{2}]$. The obtained function will start at the origin and will end at some point $(\sqrt{2},y_0)$, with y_0 (hopefully) near 0. Notice that, unlike in the first transformation process, the obtained function may take values outside the rotated square (if, for example, almost all values of the noise sequence were 1). This problem will be solved later.
4. The points (x_r,y_r) of the function are rotated 45° counterclockwise, using the following equations:

$$\begin{aligned} x &= (x_r - y_r)\cdot\sqrt{2}/2, \\ y &= (x_r + y_r)\cdot\sqrt{2}/2. \end{aligned} \tag{4}$$

5. After that, the function starts at the origin and ends at some point (x_1,y_1) on the line $y = 2-x$, hopefully near $(1,1)$. This function is strictly increasing. Then, the abscissas of all the points are divided by x_1, and the ordinates by y_1. The result will be a strictly increasing function ending at the point $(1,1)$.

In order to see which colors of noise produce the best results, several functions have been generated in this way using different noise sequences.

Figures 5, 6, 7 and 8 show encryption functions obtained from sequences of noise of the following colors:

– White noise: it has a flat frequency spectrum.
– Pink noise: its power density decreases 3 dB per octave when frequency increases.
– Red (Brownian) noise: its power density decreases 6 dB per octave.
– Noise with power density decreasing 9 dB per octave, which we decided to call infrared noise.

The white noise is the easiest to generate, since it corresponds to an uncorrelated random sequence with zero mean. In particular, we used a pseudo-random number generator to produce float numbers in the interval $[-1,1]$ following a uniform distribution.

In Figure 5 it is shown the function obtained from a white noise signal. It is clear that this kind of noise is not adequate for the generation of order preserving encryption functions. The problem is that the fast oscillations of the slope (which makes the function very unpredictable at the proximity of any point) keep the function too close to the identity.

The generation of pink noise is not trivial. One of the classical methods consists on applying a filter to a white

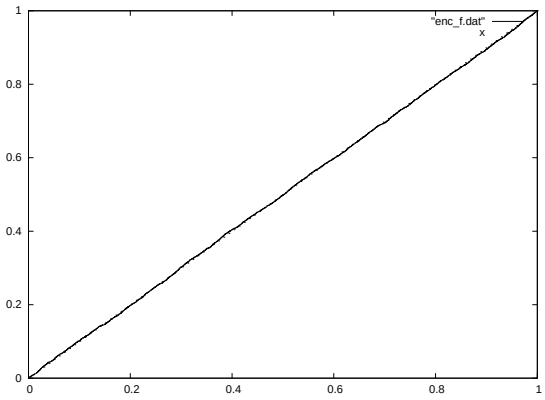


Figure 5: Encryption function generated from white noise.

noise source, but there does not exist a filter capable of producing pure pink noise from white noise, because of nonlinearity [9]. So, the result obtained is only an approximation (whose quality depends on the number of zeros/poles of the filter).

We decided to use a method inspired in [10], that offers a good noise quality at a low CPU cost. It starts from a short white noise sequence (of only two values) and follows several iterations in which the sequence length is doubled (using the mean between each pair of values) and a new white noise sequence is added. This process is repeated until the sequence has the desired length.

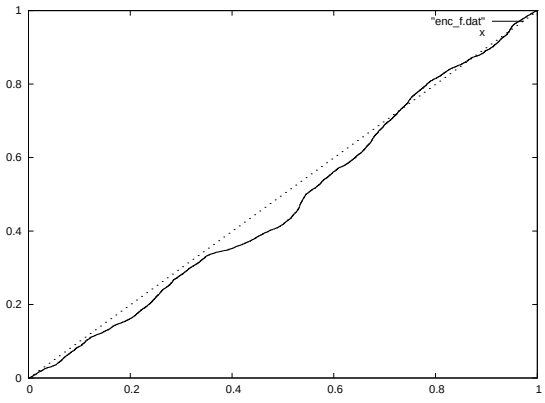


Figure 6: Encryption function generated from pink noise.

Figure 6 shows the function obtained from the pink noise signal generated in this way. It starts to diverge from the identity at some places, which makes it less globally predictable. The function has some minor autocorrelation, so it is slightly more predictable at the proximity of any point.

A red noise source can be obtained by filtering a white noise signal with a first order integrator. Each red noise sample is the sum of the first terms of the white sequence. So, they can be efficiently computed as:

$$red[k] = red[k-1] + white[k] \text{ with } red[0] = white[0]. \quad (5)$$

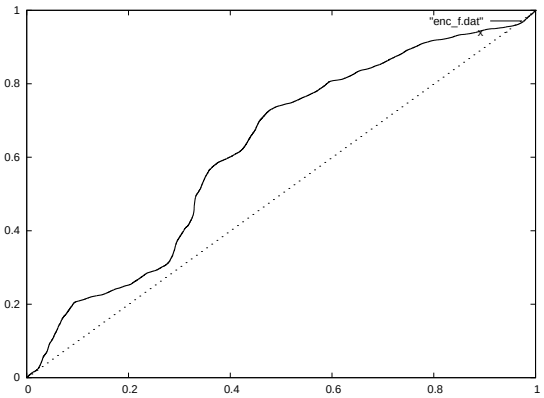


Figure 7: Encryption function generated from red noise.

The function obtained from this kind of noise (see Figure 7) is also good. It has less local divergences than the pink one, since its autocorrelation is larger. But this is compensated with less global predictability.

The same filter that produces red noise (decreasing 6 dB per octave) from a white noise source (flat, i.e. decreasing 0 dB per octave) can be used on a pink noise source (decreasing 3 dB per octave) to produce what we call infrared noise (decreasing 9 dB per octave).

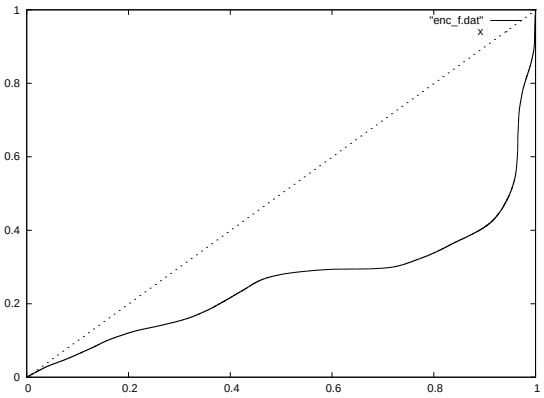


Figure 8: Encryption function generated from infrared noise.

The result, shown at Figure 8, is a function that diverges from the identity but is so smooth that, if an attacker knew some of the function points (besides the

origin and the point $(1,1)$, which are fixed), she could easily approximate the function with high precision.

Noise sequences with a power density increasing when frequency increases, like blue (azure) or violet (purple), can be obtained by differentiating pink and white noise respectively. They have also been tested, but their results were worse than the obtained with white noise.

In the same way, noise decreasing more than 9 dB per octave can be obtained by integrating red and infrared noise. The result is even smoother than the obtained in the infrared case.

So, the best order preserving encryption functions are the obtained from pink or red noise sources. These kind of noises make the function depart from the identity while maintaining an acceptable amount of randomness at the proximity of any point.

6 Security of an Encryption Function

Traditional security analysis techniques usually prove that an attacker with access to an oracle for decrypting messages in polynomial time could use it to solve, also in polynomial time, a problem believed to be hard (for example, integer factorization). So, assuming this problem is really not solvable in polynomial time, such an oracle cannot exist.

But order preserving cryptosystems are not based on hard problems. OPE encryption functions are simply strictly increasing functions, so anyone who knows some of its points (i.e. plaintext-ciphertext pairs) can approximate the encryption function (and the decryption function if the coordinates are exchanged).

The noise analysis allows to determine how are the best order preserving encryption functions in terms of their randomness. So, we could test existing OPE functions to see if the sequence obtained from the transformation is red or pink noise, and, if not, we can safely discard the cryptosystem since it will be too predictable.

Even if the analysis determines that the sequence obtained is red or pink noise, it is still possible that some functions are more predictable than others. Thus, we need a more accurate method to test the security of an OPE function.

6.1 Attack Scenario

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

Under these assumptions, 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.

Clearly, the more points she knows, the better the approximation will be. But also, less predictable functions will require more points to obtain a good approximation.

6.2 Measuring unpredictability

In order to evaluate how unpredictable is a particular OPE function, we need a metric to test how good is an approximation of the function when the attacker knows some of the points it goes through.

We initially considered using the root-mean-square error (RMSE) between the decryption function and its approximation, but it was discarded since it gives too much weight to short intervals of large errors [11]. Moreover, the RMSE does not work well as a metric of unpredictability. Notice that, since absolute errors may be considered as one-dimensional distances (which define a metric) their squares are not a metric as they do not satisfy the triangle inequality [12].

A better approach is the continuous mean absolute error (MAE) which corresponds to the average of the differences between the decryption function and its approximation over all the function domain (for which we need an integral). Since the differences can be negative we need to take their absolute values, so that positive and negative errors do not cancel out. Next expression

$$u(f_d, \hat{f}_d) = \int_0^1 |f_d(y) - \hat{f}_d(y)| dy, \tag{6}$$

shows the formula used to compute the specific unpredictability of an order preserving cryptosystem for a particular approximation. Here, $f_d(\cdot)$ is the decryption function, $\hat{f}_d(\cdot)$ is the approximation obtained from a given set of points, and $u(\cdot,\cdot)$, the unpredictability, corresponds to the mean absolute error between the decryption function and the approximation (so a larger value implies less predictive power).

Then, in order to evaluate the general unpredictability of a particular OPE function, the specific unpredictability is computed for different approximations of the function with varying numbers of known (attacked) points.

The attacked points are considered with equally spaced ordinates for simplicity (although this may not be the case in a real attack). The integral is computed by dividing the domain in a large number of intervals, taking the midpoints (which correspond to possible encrypted values), and then, for each of these values, the corresponding clear value is computed with the decryption function, the approximated one is obtained by linear interpolation between the nearest attacked points, and the absolute value of their difference, multiplied by the interval width, is added to the sum.

Cubic interpolation was also tried. In theory, this kind of interpolation should be a better option for smoother functions (like the ones obtained with reddish noises) but

care must be taken to avoid situations in which the interpolation produces decreasing intervals. In those situations, linear interpolation should be used instead. In practice, there was not a noticeable gain in predictive power.

In the worst case, when the attacker can approximate the function very well (e.g., when the encryption function is very close to the identity or when the attacker knows a large amount of points), the mean absolute error will clearly tend to zero.

In the best case, which happens when the decryption function is very degenerated (e.g., when it is very close to the unit step function, $H(y)$, with $H(0) = 0$), and the attacker only knows the two extreme points, the point $(1,1)$ and the origin (so, the approximated decryption function will be the identity, $\mathrm{id}(y)$), the MAE will tend to $1/2$:

$$\begin{aligned} u_{Max} = u(H,\mathrm{id}) &= \int_0^1 |H(y) - \mathrm{id}(y)|dy \\ &= \int_0^1 |1-y|dy = \tfrac{1}{2}. \end{aligned} \tag{7}$$

In fact, for an attack with $n+1$ known points, of the form $A_i = (x_i, y_i)$, with $A_0 = (0,0)$ and $A_n = (1,1)$, and whose ordinates are equally spaced, the maximum MAE occurs when the decryption function is a staircase whose only jumps are at the attacked points. An example of one of these functions is the following:

$$f_{d_n}(y) = \begin{cases} 0 & \text{if } y = y_0 = 0, \\ \vdots & \\ x_i & \text{if } y_{i-1} < y \leq y_i, \\ \vdots & \\ 1 & \text{if } y_{n-1} < y \leq 1. \end{cases} \tag{8}$$

Their MAEs are equal to the area of n half rectangles with height $1/n$ (since ordinates are equally spaced):

$$\begin{aligned} u_{Max_n} &= \sum_{k=1}^{n} \frac{(x_k - x_{k-1})\overbrace{(y_k - y_{k-1})}^{1/n}}{2} \\ &= \tfrac{1}{2n}\sum_{k=1}^{n}(x_k - x_{k-1}) = \tfrac{1}{2n}(x_n - x_0) = \tfrac{1}{2n}. \end{aligned} \tag{9}$$

Notice that the case $n = 1$ corresponds to the knowledge of only the two extreme points.

In fact, encryption functions achieving these maximum MAEs are not adequate, since they collapse most of the clear values to a small set of possible encrypted values (so, they have a great information loss). Moreover, each $f_{d_n}(y)$ was constructed to have the maximum MAE for an attacker knowing $n+1$ points with equally spaced ordinates. But, no encryption function achieves the maximum MAE for all possible sets of attacked points.

So, the best functions are those whose specific unpredictabilities, for most sets of attacked points, lie somewhere in between the minimum (0) and the maximum ($\frac{1}{2n}$) unpredictability, corresponding to minimum and maximum distance from the staircases. So, in order to maximize their entropy, we propose that the recommended MAEs should be the average of the two values: $\frac{1}{4n}$.

7 Experimental Results

In this section, we analyze the Noise-based Order Preserving Encryption (NOPE) scheme, as proposed in Section 5.1.

In Section 7.1, the unpredictability analysis has been used in order to evaluate the security of the different versions. Section 7.2 evaluates the efficiency of the pink Noise-based OPE scheme (pNOPE). Notice that, encryption and decryption requires the same time for any noise color, and key generation will be even faster for red or white noise (since pink noise generation is harder).

7.1 Predictability

In the following figures, each line corresponds to a particular instance of encryption method and key length. For each of these instances, we created five different keys, and, for each key, we considered several approximations of the decryption function with different amounts of attacked points.

The number of attacked points considered were 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). In each case, the points have equally spaced ordinates, and the abscissas are their decryptions.

The horizontal axis corresponds to the number of attacked points considered, and the vertical axis corresponds to the average of the MAE between five pairs of decryption function and approximation (with the corresponding encryption method, key length and number of attacked points). Both axis are in logarithmic scale.

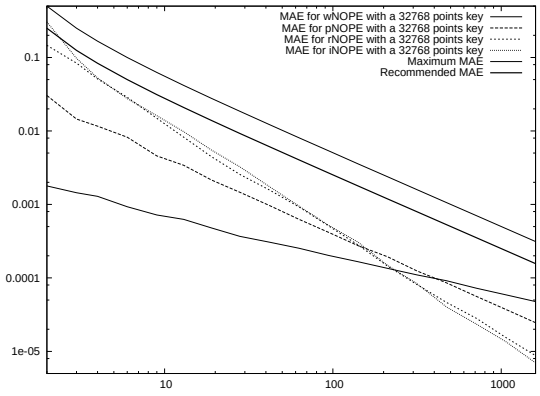


Figure 9: Mean absolute errors for the NOPE schemes.

Figure 9 shows the averaged mean absolute error for the infrared, red, pink and white Noise-based OPE schemes.

Each line (except the two higher ones) corresponds to one of the versions of the cryptosystem (i.e. the color of the noise used to generate the function), while the number of points of the key is fixed to 32768.

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, in fact, degenerated.

In the figure, the influence of the color of the noise associated to the cryptosystem can be easily appreciated. When the attacker knows a low amount of points, the 'coarse grained' randomness of the reddish noises (red and infrared) implies a better performance as the whiter ones (white and pink). But, for larger attacks, the smoothness associated with reddish noise functions make them easy to approximate, while the whiter ones retain an important amount of local randomness, that will difficult a correct approximation.

So, if we suspect that an attacker may learn a large amount of points by some means, the whiter the noise the better. Otherwise, red noise is probably more adequate, since its performance is very similar to infrared noise and is easier to generate.

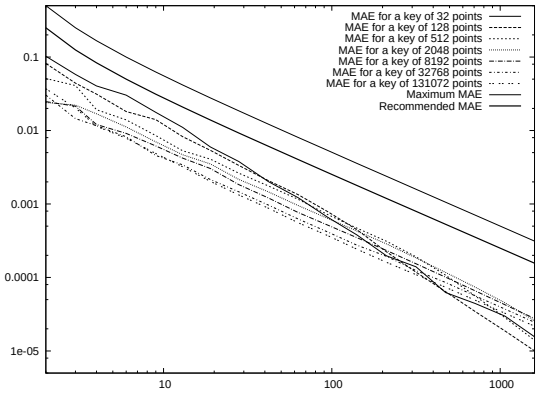


Figure 10: Mean absolute errors for the pNOPE scheme.

In Figure 10 we show the averaged mean absolute error for the pink Noise-based OPE scheme (pNOPE). In this case, each line corresponds to a particular key length (except the maximum and recommended MAE lines). Tests with more key sizes were made, but we decided to omit them for the clarity of the figure.

It can be observed that, 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.

In fact, for small attacks, functions with smaller keys perform better than the larger ones. This is due to the fact that the encryption function is a piecewise linear curve, so its changes are more abrupt when the number of defining points is lower.

But, having less defining points also means that its linear segments are longer, so if the attacker knows two correct points that lie in a particular line segment, then, as she will use linear interpolation, she will be able to correctly predict all the intermediate points.

7.2 Efficiency

In this section, we evaluate the efficiency of the pNOPE cryptosystem.

The encryption function of the pNOPE cryptosystem (and of the other versions of NOPE) consists on a list of points that the function intersects. The encryption of a value requires finding the key points whose abscissas precede and succeed the value and then interpolate between these two points to find a point whose abscissa is the point to encrypt. The ordinate of this point is the encrypted value. Considering that the key points are sorted by their abscissas (using the ordinates yields, obviously, the same order), finding the two points to interpolate can be done in logarithmic time. After that, the interpolation itself is done in constant time.

Decryption is similar; it requires finding the key points with ordinates preceding and succeeding the value to decrypt, and interpolate them to find a point whose abscissa is the decrypted value. So, its execution time is the same.

The experimentation results are shown in Table 1. It reflects the number of points of the key, its resulting size, the generation time for a key of that size, $GenT$, and the time needed to encrypt a 64 bit floating point number (type `double` in the C language), $EncT$.

Different amounts of points have been used to test both key generation and encryption/decryption. The experimentation has been performed with numbers of points of the form 2^l, for $4 \leq l \leq 26$, using a 2.4 GHz computer.

For each value of l, ten encryption functions have been generated. Each of these functions has been used to encrypt a set of one million values between 0 and 1.

Key points coordinates have also been represented as 64 bit floating point numbers, so that the size of a key, in bytes, is 16 times its number of points.

In order to obtain the generation time of the keys, column $GenT$, we computed the mean of the generation time of the ten keys of each size. To measure the encryption time of a single value, column $EncT$, we computed the mean of the encryption time of the ten data sets of each key size, divided by the number of values of each set (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

Table 1: Experimental Results of the pNOPE Cryptosystem

l	Num. Points	Key Size	*GenT*	*EncT*
4	16	256 B	19.6 μs	14.7 ns
5	32	512 B	21.5 μs	18.2 ns
6	64	1 KiB	24.6 μs	21.8 ns
7	128	2 KiB	30.7 μs	25.3 ns
8	256	4 KiB	43.1 μs	28.8 ns
9	512	8 KiB	68.2 μs	33.6 ns
10	1024	16 KiB	121 μs	38.9 ns
11	2048	32 KiB	197 μs	43.0 ns
12	4096	64 KiB	420 μs	46.9 ns
13	8192	128 KiB	854 μs	50.7 ns
14	16384	256 KiB	1.64 ms	54.7 ns
15	32768	512 KiB	3.49 ms	58.7 ns
16	65536	1 MiB	6.30 ms	62.4 ns
17	131072	2 MiB	13.0 ms	66.1 ns
18	262144	4 MiB	25.9 ms	70.3 ns
19	524288	8 MiB	53.1 ms	76.6 ns
20	1048576	16 MiB	107 ms	85.4 ns
21	2097152	32 MiB	213 ms	93.7 ns
22	4194304	64 MiB	425 ms	104 ns
23	8388608	128 MiB	848 ms	120 ns
24	16777216	256 MiB	1.63 s	141 ns
25	33554432	512 MiB	3.26 s	178 ns
26	67108864	1 GiB	6.57 s	258 ns

there are some overheads associated which increment their generation times.

Encryption times are logarithmic. And decryption times (not shown) were almost identical. So, the cryptosystem is considerably fast. For example, with a key of 65536 points ($l = 16$), the time needed for the encryption of one million values would be of about 62.4 ms ($1000000 \cdot 62.4$ ns).

As for the key length, if its size were a problem, the random seed used for its generation could be stored in its place. This way, whenever it is needed, it can be generated again (maintaining it in memory the maximum time possible, in order to reduce the number of regenerations). Obviously, the seed (or the key, if it were small) must be stored encrypted with an appropriate cryptosystem.

8 Conclusions

In this paper, two different techniques have been proposed for the analysis of order preserving cryptosystems. One of the techniques lead to the design of a new, highly efficient, cryptosystem (whose security can be improved).

The first technique consists on transforming the encryption function into a signal and then use a fast Fourier transform in order to test whether it corresponds to a noise signal or not, and the color of the noise.

This process can be reversed, so that, from a pink (or any other color) noise signal, we can obtain an order preserving encryption function. This cryptosystem is highly efficient, since encryption/decryption is logarithmic in the size of the key.

The second technique may be used to quantify the predictability of the encryption function. It computes the mean absolute error between the encryption function and several approximations obtained from the interpolation of small sets of known plaintext-ciphertext pairs.

We used both techniques to evaluate the security of the new Noise-based Order Preserving Encryption methods (NOPE). An experimentation was also performed, in order to test the pink NOPE (pNOPE) speed. It confirmed the expected results.

When a database query is made by a legitimate user, the database manager will be responsible for encrypting the necessary values to answer the query, and, after that, decrypting the values to return. So, normally, users have no access to encrypted data.

If an attacker got access to order-preserving encrypted values, the amount of information she could obtain will depend on the randomness of the encryption function and on the number of plaintext-ciphertext pairs she knew beforehand (if any). One of the reasons why she might know some pairs is that she knew the correct value for some records and then, upon accessing the internal data, she could relate them to the encrypted data.

For avoiding this problem, it is highly recommended that all the database fields are encrypted, using one cryptosystem or another depending on the type of queries that should be permitted. Moreover, if the database contains multiple sortable fields, each of them should be encrypted 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] James Groff and Paul Weinberg. *SQL The Complete Reference, 3rd Edition*. McGraw-Hill, Inc., 2010.

[2] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-Preserving Symmetric Encryption. In *28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques - EUROCRYPT'09*, pages 224–241, 2009.

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

[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.

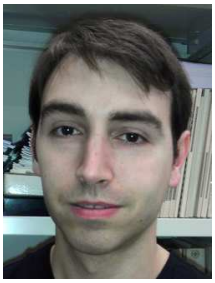
[8] Vittorio Bagini and Marco Bucci. A Design of Reliable True Random Number Generator for Cryptographic Applications. In *Cryptographic Hardware and Embedded Systems*, pages 728–728. Springer, 1999.

[9] S. Plaszczynski. Generating long streams of $1/f^{\alpha}$ noise. *Fluctuation and Noise Letters*, 7(1):R1–R13, 2007.

[10] Alain Fournier, Don Fussell, and Loren Carpenter. Computer Rendering of Stochastic Models. *Communications of the ACM*, 25(6):371–384, June 1982.

[11] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1):79–82, December 2005.

[12] Paul W. Mielke Jr. and Kenneth J. Berry. *Permutation Methods: A Distance Function Approach*. Springer Series in Statistics. Springer-Verlag, New York, 2001.



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, Spain, in 2002 and 2004, respectively, and his Ph.D. in Engineering from Universitat de Lleida, Spain, 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, Spain, in 1983, and his Ph.D. in Mathematics from Universitat Politècnica de Catalunya, Spain, 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 a former postdoctoral research assistant at Universitat de Lleida. She received her B.Sc. in Computer Science from Universitat de Lleida, Spain, in 2002, her M.Sc. in Computer Science from Universitat Rovira i Virgili, Spain, in 2004, and her Ph.D. in Engineering from Universitat de Lleida, Spain, 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, Spain, 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.