Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 891–899
DOI: 10.15439/2014F186
ACSIS, Vol. 2

Order-preserving encryption schemes based on arithmetic coding and matrices

Sergey F. Krendelev
Lab of Modern Computer Technologies, Novosibirsk State University, Novosibirsk, Russia
Email: s.f.krendelev@gmail.com

Mikhail Yakovlev
Dept. of Information Technology, Novosibirsk State University, Novosibirsk, Russia
Email: m.o.yakovlev@gmail.com

Maria Usoltseva
Dept. of Information Technology, Novosibirsk State University, Novosibirsk, Russia
Email: m.a.usoltseva@gmail.com

Abstract—**In this article we describe two alternative order-preserving encryption schemes. First scheme is based on arithmetic coding and the second scheme uses sequence of matrices for data encrypting. In the beginning of this paper we briefly describe previous related work published in recent time. Then we propose alternative variants of OPE and consider them in details. We examine drawbacks of these schemes and suggest possible ways of their improvement. Finally we present statistical results of implemented prototypes and discuss further work.**

I. Introduction

Security is a fundamental issue solved by DBMS and cloud service designers. Using cryptographic algorithms to store confidential data in encrypted from isn't always the best solution. In case of relational (SQL) database it is impossible to process encrypted data on DBMS side. Generally two options are possible. The first one is to decrypt and process data on client side, which leads to significant traffic increase between DBMS and application due to general inability to single out the necessary data. Second option is to decrypt data on DBMS side, which is unsafe in case DBMS or cloud service isn't reliable.

The research speculates on ability to use special types of encrypting, which allow not only to safely store data, but also to perform a set of operations on it. Particularly, the research concerns order-preserving cryptosystems.

Definition (order-preserving function). Let A, B be sets with given order relation on each set. Function

$$F : A \to B$$

is said to preserve order if

$$x < y \ (x, y \in A) \Leftrightarrow F(x) < F(y).$$

Encryption based on using such functions is called order-preserving encryption (OPE). Assume there is a set of unique plaintexts $P = p_1, p_2, \dots p_{|p|}, p_i < p_{i+1}$. The corresponding encrypted values are represented as $C = c_1, c_2, \dots c_{|c|}, c_i < c_{i+1}$. Such encryption enables subset of SQL operations on encrypted data including, e.g. selection of encrypted values intervals.

II. Related work

In the last 10 years several scientific papers has published, which introduced different schemes of order-preserving encryption.

A. OPE based on pseudo-random number generator

One of the first approaches, represented in the research [1], suggests that integer number p is encrypted as follows:

$$c = \sum_{j=0}^{p} R_j,$$

where R_j is jth number generated by reliable pseudo-random generator. The drawbacks of this scheme are the memory footprint of encrypted values and possible overflow, resulting from calculation of ciphertexts for large plaintexts while working with built-in types. There is also the complexity of adding new plaintexts: for adding p_i where $p_i < p < p_{i+1}$ we need to re-encrypt values $p_j, j > i$.

That’s why this method is ineffective for big numbers and, in some cases, the encryption result can be predicted. For instance, suppose μ average distribution of numbers, generated by the pseudo-random number generator. For uniform distribution on the interval $[1 \dots Max]$, $\mu = \frac{Max+1}{2}$, then $f(x) = \mu x$ will approximate encryption function.

B. OPE based on polynomial functions

In research [2] a sequence of strictly increasing polynomial functions is used for encrypting integer values. These polynomials may be of first or second order. The secret key is polynomial coefficients. Sequence of polynomials is applied to initial number in a way that one function's output value is another function's input value. Decryption is done by applying inverse functions in reverse order. Sometimes it might be impossible to find the inverse function for a specific polynomial. Authors suggest using simple polynomials $f(x) = ax^b + c$ as they all have inverse

This research was performed in Novosibirsk State University under support of Ministry of education and science of Russia (contract no. 02.G25.31.0054).

functions $f(x)=\sqrt[b]{\frac{x-c}{a}}$. Besides, maximum degree of such polynomials, according to the authors, does not exceed 2, and a set of possible coefficients – $\{1...32\}$. In this case decryption is a lot more complicated than encryption due to square root operation. As built-in integer types are implemented with fixed length, inevitable overflow errors appear. In this work authors suggest using logarithmic functions $f(x)=\log_2 x+c$. This implies working with non-integer types, therefore accuracy errors should be considered. To avoid accuracy and overflow errors this scheme requires rather complex selection of parameters.

C. Research by R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu

Research [3] speculates on encryption of data from subset of integers, authors also think it possible to use non-integer types represented as integers of the same size. This method encrypts data so that ciphertexts follow a certain distribution selected by the user. In order to generate encryption function they use all data that needs to be encrypted (if the database doesn't contain records, administrator has to add several possible records) and the list of possible sample distributions. Key is generated from all these samples. Besides, in order to simulate distribution data has to be split into buckets, in which linear interpolation is used. One of the main drawbacks is that the time of key generation time is linear in the size of database. And if the key has already been generated with adding of new records it might need to be regenerated along with re-encryption of data.

D. Research by A. Boldyreva N. Chenette, Y. Lee and A. O'Neill

A. Boldyreva in [5], [6] shows the connection between OPE schemes and hypergeometric (HG) и and negative hypergeometric (NHG) distributions. The connection allows to simulate OPE scheme through HG or NHG generator. There are effective algorithms of accurate random value generation of these distributions. One of order-preserving encryption features is that range of encryption function is always larger than input argument set (two different numbers can't be encrypted to the same numbers). Suppose encryption function maps a set $[1...M]$ to a set $[1...N]$, $N>M$. In order to encrypt the set of numbers $[1...M]$ we need M random numbers from the set $[1...N]$. Let us denote the set of these numbers by N_M. They have to be ordered and associated correspondingly with initial numbers. I.e. function $f:[1...M]\rightarrow N_M$ maps the ith element of the set $[1...M]$ to the ith element of the set N_M. Since we don't know, how many plaintexts have to be encrypted, we cannot determine the size of the set $[1...N]$. Authors offer to generate the next element of the ordered set N_M only when it is necessary for ciphertext generation. This approach is called lazily sampling the function.

Let m be a plaintext from the set $[1...M]$, g – a function, generating the set $[1...N]$, x – number of elements selected into the set N_M after y steps. The number x is characterized by the hypergeometric distribution. Encryption starts with entire domain $[1...M]$ and range $[1...N]$. Let $y=\frac{\max(N)}{2}$ be a range gap. With a certain key and initial parameters the number x can be calculated. If $x<m$, we need to consider the points of the domain greater x and y, and less than or equal to x and less than or equal to y in case of $x\ge m$. As a result we get admissible set of ciphertexts.

E. Alternative OPE schemes

In our research we consider other ideas, which OPE can be based on. We propose two alternative OPE schemes, research problems of overflow and computational accuracy and try to increase cryptographic strength of schemes.

III. OPE Scheme Based on Arithmetic Coding

This scheme builds a representation of integer in a suitable form. Representation preserves the order, so we can talk about order-preserving mapping. Suppose we consider positive integers requiring for their representation not more than n bits. Each number c is mapped to a bit string $b=(a_1,a_2,...a_n)$, where a_1 is MSB. Let us define the order-preserving mapping f. We assume that the string defines certain real number $s\in[0,1)$. The simplest way to define it:

$$s=a_1\frac{1}{2}+a_2\frac{1}{2^2}+\cdots+a_n\frac{1}{2^n}.$$

In other words $s=\frac{c}{2^n}$.

Let us seek a different representation of the number s. To do it let us use ideas associated with arithmetic coding. The equation:

$$G(x)=2^n x-c=0 \qquad (1)$$

has only one solution on the interval $[0,1)$. In case of bisection method of solving the equation (1) we get the number s after n steps. The main idea of arithmetic coding is that segments can be split into parts arbitrarily. In this case approximate solution of equation (1) can be found in fewer steps. That allows us to achieve compression of data while using arithmetic coding.

Let us describe the process of segments splitting. Suppose, $\gamma=\frac{p}{p+q}$, $\mu=\frac{q}{p+q}$, where p, q are random natural numbers. Obviously, $\gamma+\mu=1$. Let us split segment $[0;1)$ into two parts $\left[0,\frac{p}{p+q}\right)$, $\left[\frac{p}{p+q},1\right)$. After that, if $G\left(\frac{p}{p+q}\right)>0$, we choose $\left[0,\frac{p}{p+q}\right)$ and produce 0. In case of $G\left(\frac{p}{p+q}\right)<0,$ we choose $\left[\frac{p}{p+q},1\right)$ and produce 1. Let us denote the segment we chose by $[a_1,b_1)$.

New segment splits into parts in the ratio $\gamma:\mu.$ According to the sign of $G(x)$ in the point, one of the segments is selected. Let us denote it by $[a_2,b_2)$. Proceeding by induction, we compute the interval $[a_n,b_n)$. Its length is $\gamma^r\mu^{n-r}$, where r is number of zeros in b. By construction of $b_n, b_n \in Q,$ so it can be expanded in powers of $\frac{1}{2}$ up to m degree, where m is the smallest integer satisfying

$$\frac{1}{2^m}<\gamma^r\mu^{n-r}.$$

That condition may also be written as

$$-m<r\log_2\gamma+(n-r)\log_2\mu$$

or

$$m>-r\log_2\frac{p}{p+q}-(n-r)\log_2\frac{q}{p+q}=r\log_2\frac{p+q}{p}+$$
$$+(n-r)\log_2\frac{p+q}{q}=n\log_2(p+q)-r\log_2 p-$$
$$-(n-r)\log_2 q$$

Therefore, we can calculate m, if we know $b,\gamma,\mu.$ For universality we need m estimation not to depend on r value. E.g, we can require condition $m>n\log_2(p+q)$. Let us approximate b_n with bit string $f(b)=(\beta_1,\beta_2,\dots,\beta_n)$. Obviously, this transformation preserves order.

General conclusion, which is using adaptive arithmetic coding, is to use different ratio on each step, when approximating the solution of equation (1). It allows making cryptosystem cryptographically stronger.

VI. Key generation

Key is set of ratios, which segments are divided in. Suppose current segment is split in a ratio $p_i:q_i$ on ith step. To be able to decrypt the cipher for n-bit number, the length of the segment obtained after decryption should be less than $\frac{1}{2^n}$. Maximum segment length, which can be obtained as a result of decrypting is $\prod_i\frac{\max(p_i,q_i)}{p_i+q_i}$. Consequently, the key generation algorithm has the following form:

1. Choose random ratio $p_i:q_i$.
2. Check the condition

$$\prod_i\frac{\max(p_i,q_i)}{p_i+q_i}<\frac{1}{2^n}.$$

If condition is satisfied, go to step 3, otherwise – to 1.

3. Complete key generation.

V. Encryption

Suppose we need to encrypt $Num\in\mathbb{N}$. On each algorithm iteration the interval $[a_i,b_i)$ is considered, where $a_0=0, b_0=1.$ Let us examine the ith iteration of the algorithm.

Current segment $[a_{i-1},b_{i-1})$ is divided in a ratio $p_i:q_i$. Let point $s\in[a_{i-1},b_{i-1})$ separate it, i.e. $s=a_{i-1}+\frac{(b_{i-1}-a_{i-1})p_i}{p_i+q_i}$. If $s>\frac{Num}{2^n}$, then the output is 0, $a_i=a_{i-1}, b_i=s$. Otherwise the output is 1, $a_i=s, b_i=b_{i-1}$. It can be seen, that $\forall i\ \frac{Num}{2^n}\in[a_i,b_i)$ (by a_i and b_i selection).

After performing k iterations (where k is key size, i.e. the number of ratios) we obtain a bit sequence $\sigma=(\sigma_1,\dots\sigma_k), \sigma_i\in\{0,1\}$, which is ciphertext for Num.

VI. Decryption

Suppose there is a bit sequence $\sigma=(\sigma_1,\dots\sigma_k), \sigma_i\in\{0,1\}$, which is a ciphertext for unknown number Num. Just as in the encryption algorithm, at each iteration is considered the interval $[a_i,b_i)$, $a_0=0, b_0=1.$ Consider the ith iteration of the algorithm.

Current segment $[a_{i-1},b_{i-1})$ is divided in a ratio $p_i:q_i$. Let point $s\in[a_{i-1},b_{i-1})$ separates it, i.e. $s=a_{i-1}+\frac{(b_{i-1}-a_{i-1})p_i}{p_i+q_i}$. If $\sigma_i=0$, then $a_i=a_{i-1}, b_i=s$. Otherwise, $a_i=s, b_i=b_{i-1}$.

After performing k iterations (where k is key size) we obtain a segment $[a_k,b_k)$, and $(b_k-a_k)<\frac{1}{2^n}$ because of key selection. As $\frac{Num}{2^n}\in[a_k,b_k)$ number Num is uniquely decoded as follows:

$$Num=\lfloor 2^n a_k\rfloor+1,$$

where $\lfloor x\rfloor$ is the largest integer, which comes before x.

VII. Increasing cryptographic strength of the algorithm

If attacker does not know a secret key, all he knows is that encrypted value belongs to the interval $[a_0,b_0)$. But to make algorithm more secure, this segment can be hidden.

Let us choose an arbitrary strictly increasing function $f(x)$ so that $\lim_{x\to-\infty}f(x)<0, f(x)=+\infty.$ Suppose $f(a)=0, f(b)=2^n.$ Let us use function $f(x)$ to encrypt n-bit number $s:0\le s<2^n.$ Using modified arithmetic encryption algorithm, we encrypt number x from interval $[a,b)$, where $f(x)=s$ (fig.1).

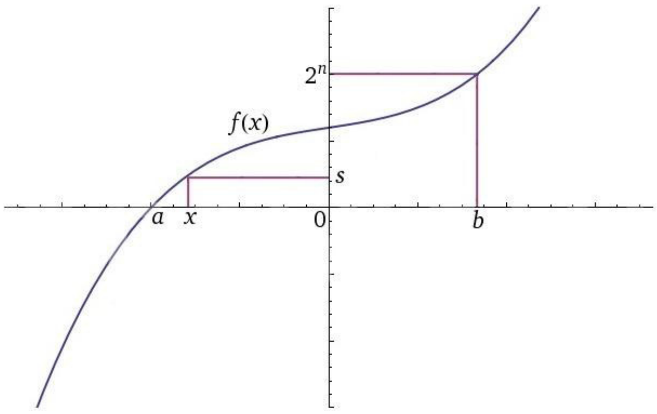

Fig.1 Increasing cryptographic strength of algorithm

In this case secret key is set of ratios p_i, q_i (key for modified arithmetic algorithm we already described earlier) and function $f(x)$, that was used in encryption.

To decode a cipher h we should find a and b such that $f(a)=0$ and $f(b)=2^n$ (they are unique, because $f(x)$ is strictly increasing function), decode x from interval $[a,b)$ and calculate $s=f(x)$. Now if attacker doesn't know a secret key, he doesn't have any information about s.

VIII. Implementation features

The algorithm described above can be implemented using only integers, which allows the implementation to eliminate rounding or computation errors. Nevertheless, the size of the numerator and denominator of fractions a_i, b_i and $x=\frac{\text{Num}}{2^n}$ can't be limited in general case, and therefore arbitrary-precision integer arithmetic should be used. As a result algorithm speed is not high enough.

The algorithm can be modified in order to accelerate implementation. On each iteration of the encryption transformation matrix $[a_i,b_i)\to[0,1)$ can be applied to segment $[a_i,b_i)$. Then it is obvious that $a_i\to 0, b_i\to 1$. Point x transforms by the following rule:

$$\text{if } \sigma_i=0, \text{then } \frac{\text{Num}}{2^n}\to\frac{\text{Num}}{2^n}\frac{p_i+q_i}{p_i}, \qquad (2)$$

$$\text{else } \frac{\text{Num}}{2^n}\to\frac{\text{Num}}{2^n}\frac{p_i+q_i}{q_i}. \qquad (3)$$

It saves time for the calculation of the segment, and long arithmetic is only required for the storage of a fractional number x.

As a result the encoding iteration includes comparing points x and $\frac{p_i}{p_i+q_i}$ and following x recalculation according to the rules (2) and (3). In case of decrypting we need to know final segment $[a_k,b_k)$, so decoding iteration is implemented directly with a_i and b_i recalculation.

In order to remove one of the slowest parts of the implementation - arbitrary-precision integer arithmetic, we can apply rounding to the largest previous integer, which is used in many implementations of standard arithmetic coding[1].

The idea of this approach is that all fractional numbers such as a_i, b_i and x, which belong to the interval $[0,1)$ are multiplied by 2^r, where r is some power of two. Thus, arithmetic operations can be quickly performed. E.g. in case of 32-bit architecture, it is convenient to choose $r=32$. After multiplying the resulting fraction shall be rounded down to the nearest integer. The more is r, the smaller is rounding error, so it is advisable to choose large values for r.

On each iteration of the encoding (or decoding) algorithm initial segment $[a_0,b_0)=[0,2^r)$ decreases, because $[a_i,b_i)\subset[a_{i-1},b_{i-1})$, so rounding error increases and at some point it is impossible to determine if $x=\left\lfloor\frac{\text{Num}}{2^n}2^r\right\rfloor$ belongs to segment $[a_{i-1},s)$ or $[s,b_{i-1})$. To avoid it, the length of interval $[a_{i-1},s)$ and $[s,b_{i-1})$ after the process of rounding should be not less than 1. In other words,

$$s-a_{i-1}\ge 1,$$

$$\frac{(b_{i-1}-a_{i-1})p_i}{p_i+q_i}\ge 1,$$

$$b_{i-1}-a_{i-1}\ge\frac{p_i+q_i}{p_i}.$$

Similarly for the interval $[s,b_{i-1})$:

$$b_{i-1}-a_{i-1}\ge\frac{p_i+q_i}{p_i}.$$

Thus, in order to be able to perform the ith iteration, the current length of the segment $[a_{i-1},b_{i-1})$ should not be less than

$$\max\left(\frac{p_i+q_i}{p_i},\frac{p_i+q_i}{q_i}\right)\le p_i+q_i\le\max_i(p_i)+\max_i(q_i) \qquad (4)$$

There is a special renormalization operation, which allows to increase the length of the current segment. It can be used in one of three cases:

1. Segment $[a_{i-1},b_{i-1})$, lies in the left half of the interval $[0,2^r)$, i.e. $[a_{i-1},b_{i-1})\subseteq[0,2^{r-1})$. In this case interval $[2^{r-1},2^r)$ is not involved in the encryption process anymore, so we can "bring closer" $[0,2^{r-1})$ segment twice, i.e. use transformation $[0,2^{r-1})\to[0,2^r)$. Then

[1]Moffat, Alistair. ACM Transactions on Information Systems / Alistair Moffat, Radford M. Neal, Ian H. Witten // – 1998. – Vol. 16, No. 3, July 1998.

$$a_{i-1} \to 2a_{i-1},$$
$$b_{i-1} \to 2b_{i-1},$$
$$x \to 2x.$$

2. Interval $[a_{i-1}, b_{i-1})$ lies in the right half of the interval $[0, 2^r)$, i.e. $[a_{i-1}, b_{i-1}) \subseteq [2^{r-1}, 2^r)$. Similarly to the previous case, we can use transformation $[2^{r-1}, 2^r) \to [0, 2^r)$. Then

$$a_{i-1} \to 2(a_{i-1} - 2^{r-1}),$$
$$b_{i-1} \to 2(b_{i-1} - 2^{r-1}),$$
$$x \to 2(x - 2^{r-1}).$$

3. Interval $[a_{i-1}, b_{i-1})$ lies in the central part of the interval $[0, 2^r)$, i.e. $[a_{i-1}, b_{i-1}) \subseteq [2^{r-2}, 3\cdot 2^{r-2})$. Then let's "bring closer" twice the interval $[2^{r-2}, 3\cdot 2^{r-2})$, i.e. use transformation $[2^{r-2}, 3\cdot 2^{r-2}) \to [0, 2^r)$. Then

$$a_{i-1} \to 2^{r-1} - 2(2^{r-1} - a_{i-1}),$$
$$b_{i-1} \to 2^{r-1} - 2(2^{r-1} - b_{i-1}),$$
$$x \to 2^{r-1} - 2(2^{r-1} - x).$$

It is easy to notice that each case of the renormalization increases the length of the segment twice. If none of the renormalization conditions is satisfied, then the length of this interval is strictly greater than 2^{r-2}. According to this we can find the maximum size of p_i and q_i. If they are m-bit numbers, then (by (4)) current segment length after renormalization should be greater than

$$\max_i(p_i) + \max_i(q_i) \le 2^m + 2^m = 2^{m+1}.$$

Then $m_{max} + 1 = r - 2$, and $m_{max} = r - 3$. Larger size makes no sense to choose, because rounding leads to (r-3)-bit p and q numbers.

As a result we turn from calculations with fractional numbers with unlimited numerator and denominator to calculations of the fixed integers. This fact significantly increases the speed of algorithm. The price of this acceleration is a rounding error, which may reduce the strength of cryptographic algorithm.

IX. Statistics

Any order-preserving encryption can be represented as transformation from domain $[0, 2^{b_1})$ to some range $[0, 2^{b_2})$, where b_1, b_2 are the sizes of input and output data respectively, so it is possible to attack a cipher using linear approximation cryptograms on extreme values (fig 2).

In other words, cryptogram for a certain $x \in [0, 2^{b_1})$ is approximated with value $2^{b_2 - b_1} \cdot x$, and plaintext for cryptogram $m \in [0, 2^{b_2})$ – with value $\left\lfloor \frac{m}{2^{b_2-b_1}} \right\rfloor$. Knowledge of the secret key is not required to calculate the approximate value, only sizes of the input and output data are needed. To defend against such attack we should increase an approximation error.

Therefore, in order to study the cryptographic strength of the algorithm statistics of "segment" lengths was examined. The end points of such "segment" a_i equals successive encrypting function values, i.e. $a_i = [f(i), f(i+1)]$. Obviously, the closer "segment" lengths distribution to uniform distribution, the smaller the error of approximation.

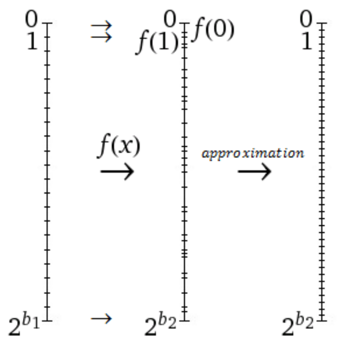

Fig.2 Linear approximation of cryptograms on extreme values

As a result, it was found that the "segment" lengths distribution is close to exponential (fig. 3, the abscissa indicates the segments lengths and the ordinate indicates the number of such segments). As too big error appears, this attack cannot be applied.

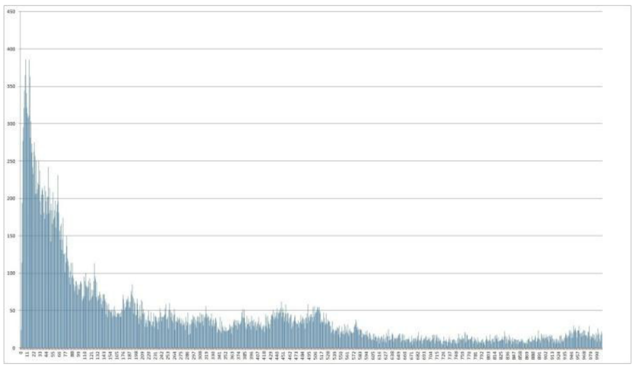

Fig.3 "Segment" lengths distribution of two-byte numbers

X. Matrix based OPE scheme

The suggested scheme allows to avoid overflow accuracy errors as it uses only integers. The scheme does not disclose any information about initial values of encrypted variables, except their order.

Tuple of three numbers (r, k, t) serves as a ciphertext. Suppose we need to encrypt positive integer x. In contradistinction to pseudo-random number generator method the first element of cryptogram is not the sum of numbers of random ascending sequence on the step x, but a number i of a step, such that sum of random numbers exceeds the initial number x on this step. The second element of the cipher is the difference between x and the sum of random sequence on (i-1)th step. It can also be encrypted in a similar way, using the sum of random elements of a different sequence

$$g_1 < g_2 < \ldots < g_{r-1} \le x < g_r .$$

Obviously, $x - g_{r-1} < g_r - g_{r-1}$. Therefore, to encrypt residual $x - g_{r-1}$ we need to calculate the sum of the sequence elements, which will be limited by $g_r - g_{r-1}$. Instead of pseudo-random sequence elements sum we can use strictly increasing function with special features. The sum of the sum sequence of matrix elements is suggested to be used as strictly increasing function:

$$\sum_{i=0} \sum_{j=0}^{n} a_i ,$$

where a_i is an elements of matrix A_i of size $n \times n$. We use a certain sequence of matrices

$$A_1, A_2, \ldots, A_r, \ldots$$

First, a certain matrix A_1 is chosen. Each successive matrix can be calculated using some transformation of the previous one. As such transformation we research power function.

$$A_1, A_1^2, \ldots, A_1^r, \ldots$$

In order to prevent the increasing of matrix elements after multiplication operation, we consider matrix elements are from finite field $\mathbb{Z}_m$. Residual $x - g_{r-1}$ is hidden with the help of elements of the matrix, calculated on the rth step. Elements are summed up randomly, and the number k of step, where the sum s_k exceeds $x - g_{r-1}$, is the second element of the tuple. The residual obtained at this stage is the third element of the tuple. Let's look at the scheme in greater detail.

Encryption scheme under consideration is symmetrical, i.e. it uses private key for encryption and decryption of data. In the research the input data of cryptographic algorithm are integer non-negative numbers from 0 to 2^n, where $n = 16, 32, 64$, which corresponds to the size of built-in types unsigned short int, unsigned int, unsigned long int, unsigned long long int of C++ language, which the scheme is implemented on. The result of cryptographic algorithm is a tuple of three numbers (r, k, t), where $r, k, t \in \mathbb{Z}_{\ge 0}$.

XI. Key generation

The private key in the scheme is non-degenerate matrix $A(\det A \neq 0)$ over the finite field $\mathbb{Z}_m$, $m \ge 2$, with $n \times n$ size, where $n \in N, n \ge 2$ and a certain permutation of σ elements of matrix with such size, called matrix traversal.

A. Non-degenerate matrix generations

Standard way of non-degenerate matrix generation is to generate matrix with random elements and check its non-degeneracy. If the determinant of obtained matrix is equal to zero, generation shall be repeated until this condition is violated.

For small-size matrices this method is quite effective. However with increase of matrix size, multiplication operation takes more and more time, which reduce speed of private key generation.

Direct methods of determinant calculation can be based on permutations sum, or Laplace expansion of smaller degree determinants. However such methods are rather ineffective, because they require time complexity $O(n!)$ for n degree determinant calculation. There are other methods with fewer operations, e.g. Gauss method modification, where matrix is transformed to the form of echelon and determinant is calculated as product of multiplication of diagonal elements. Complexity of this method constitutes $O(n^3)$. If there is available multiplication algorithm of two square matrices of size n in time $M(n)$, where $M(n) \ge n^a$, for certain $a > 2$, then matrix determinant can be calculated in time $O(M(n))$. However if random matrix is degenerate, calculations shall be repeated until we get non-degenerate matrix.

This research suggests using knowingly non-degenerate matrix generation approach, based on the following linear algebra theorem.

Theorem. Square matrix A with non-zero principal minors can be presented in the form of LU lower triangular matrix L, whose main diagonal consists of non-zero elements, times upper triangular matrix U with units on the main diagonal.

Since lower triangular matrix L contains a single diagonal, its determinant equals one. Upper triangular matrix U determinant equals multiplication of elements on main diagonal. Using the property of determinant, we obtain $\det(A) = \det(LU) = \det(L)\det(U) = \det(U)$.

Thus, in order to generate non-degenerate matrix A, it is enough to generate matrices L and U, compliant with the above properties and find their multiplication result. Computational complexity of matrix product by definition is $O(n^3)$, however there are more effective algorithms, for instance, Coppersmith–Winograd algorithm, performing multiplication in $O(n^{2.3727})$.

B. Generation of matrix elements permutation

Algorithm uses permutation of matrix elements, imitating its traversal, a specific order in which to trace the elements of a matrix. Cryptographic robustness cannot be achieved through simple row traversal of matrix elements or row traversal with a shift. The traversal should be generated randomly. Traversing function shall go over all matrix elements (through each element only once), i.e. the function has to be bijective. For instance, we can use affine transformation.

It shall be presented through randomly generated non-degenerate in $\mathbb{Z}_m$ matrix B of size *2×2* (B shouldn't be identity matrix, or else the function implements row traversal with a shift) and vector $c = (c_1, c_2)$, where $c_1, c_2 \in \mathbb{Z}_m$. Matrix non-degeneracy provides inverse mapping, used in decryption. New element coordinates are calculated as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix} = B\begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$

where coordinates (i, j) mean initial row traversal $(1,1), (1,2), \dots (2,1), (2,2), \dots (n,n)$ and $(u(i,j), v(i,j))$ is traversal, used in algorithm.

Generally, to determine matrix traversal we can use any effective algorithm for random permutation of set of elements generation, e.g. Fisher–Yates shuffle algorithm (Knuth shuffle), with time complexity reduced to $O(n)$, along with its updated versions – Durstenfeld and Sattolo algorithms, using less memory space. When using high quality unbiased random numbers generator, algorithm guarantees equiprobability of permutations.

XII. Encryption

Let's consider data encryption procedure. Suppose we need to encrypt $x \in \mathbb{Z}_{\geq 0}$. At first private key (A, σ) is generated. The first element r of ciphertext is calculated as follows:

$$\sum_{i=1}^{r-1} d(A^i) \leq x < \sum_{i=1}^{r} d(A^i),$$

where

$$d(A) = \sum_{i=0}^{n^2} a_i$$

is the sum of elements a_i of matrix A. Power operation A^i is performed in $\mathbb{Z}_m$.

In order to determine the second element k of the cipher, we need to calculate the sum

$$S = \sum_{i=0}^{k} a_i^{'},$$

where $a_i^{'}$ are elements of matrix σA^r, such that

$$S \leq x - \sum_{i=1}^{r-1} d(A^i).$$

Difference

$$t = x - \sum_{i=1}^{r-1} d(A^i) - S$$

is the third element of the cipher.

XIII. Decryption

Suppose the input of decryption algorithm is a tuple (r, k, t) and the key is (A, σ). First, we shall calculate matrix A^r. Using known permutation σ of matrix elements, we calculate

$$S = \sum_{i=0}^{k} a_i^{'},$$

where $a_i^{'}$ are elements of matrix σA^r. Adding to the sum S the third element of ciphertext t, we obtain a certain number h, which, according to encryption procedure, equals

$$h = t + S = x - \sum_{i=1}^{r-1} d(A^i).$$

Using the first element r of ciphertext we calculate the sum

$$\sum_{i=1}^{r-1} d(A^i)$$

and obtain the number x, that was encrypted

$$x = h + \sum_{i=1}^{r-1} d(A^i).$$

XIV. Constructinon correctness

First of all we have to verify cipher uniqueness. Thus, the encrypted number should be the same number as decrypted. We also shall prove that such encryption is order-preserving.

Theorem 1. If with encrypted N there is cipher Enc(N), then with decryption Dec(Enc(N)) there is number N.

Proof. We need to prove that mapping, determining encryption algorithm, is bijective. In bijective mapping every element of one set corresponds to only one element of another set, along with inverse mapping with the same property.

Bijective mapping properties:

1. A function $f : A \to B$ is bijective if and only if it is invertible, that is, there is a function $g : B \to A$ such that $g \circ f$ is identity function on A and $f \circ g$ is identity function on B.
2. The composition $g \circ f$ of two bijections is again a bijection.

Function $f_1(x, A)$ calculates the parameter r. Since $\det A \neq 0$, then function

$$\sum_{i=1}^{j} d(A^i)$$

is strictly increasing, i.e.

$$\sum_{i=1}^{j} d(A^i) < \sum_{i=1}^{j+1} d(A^i).$$

Obviously, for any x from $\mathbb{Z}_{\geq 0}$ number r is found uniquely. Function $f_2(x, A, \sigma)$ calculates the parameter k. Since

$$x < \sum_{i=1}^{r} d(A^i),$$

hence we get

$$x - \sum_{i=1}^{r} d(A^i) < d(A^r).$$

Therefore, there is the only k, such that

$$S = \sum_{i=0}^{k} a_i^{'}, \quad S \leq x - \sum_{i=1}^{r-1} d(A^i),$$

where $a_i^{'}$ are elements of matrix σA^r. It is obvious that permutation of matrix A^r elements does not violate this condition. If functions

$$\sum_{i=1}^{r-1} d(A^i)$$

and S are bijective, then function $f_3(x, A, \sigma)$, calculating t, is also bijective.

Theorem 2. If with encryption of N_1 with the key K we obtained cipher Enc_1, and with encryption N_2 with the same key we obtained cipher Enc_2, and $N_1 > N_2$, then $Enc_1 > Enc_2$.

Proof. Let us say that tuple $\overline{A}_1 = (a_{11}, \ldots, a_{1m+1})$ is greater than vector $\overline{A}_2 = (a_{21}, \ldots, a_{2m+1})$, if $a_{1j} > a_{2j}$, where j is first position number, such that $a_{1j} \neq a_{2j}$, for all j, from m+1 to 0.

We shall consider $x_1, x_2 \in \mathbb{Z}_{\geq 0}$, such that $x_1 < x_2$. According to encryption procedure

$$\sum_{i=1}^{r_1-1} d(A^i) \leq x_1 < \sum_{i=1}^{r_1} d(A^i),$$

$$\sum_{i=1}^{r_2-1} d(A^i) \leq x_2 < \sum_{i=1}^{r_2} d(A^i).$$

Consequently,

$$\sum_{i=1}^{r_1-1} d(A^i) \leq \sum_{i=1}^{r_2-1} d(A^i), \quad \sum_{i=1}^{r_1} d(A^i) \leq \sum_{i=1}^{r_2} d(A^i).$$

These functions are strictly increasing, so $r_1 \leq r_2$. As

$$S_1 \leq x_1 - \sum_{i=1}^{r_1-1} d(A^i),$$

$$S_2 \leq x_2 - \sum_{i=1}^{r_2-1} d(A^i),$$

therefore, $S_1 \leq S_2$. By definition of S function, since all the matrix elements are non-negative, then $k_1 \leq k_2$.

$$t_1 = x_1 - \sum_{i=1}^{r_1-1} d(A^i) - S_1,$$

$$t_2 \leq x_2 - \sum_{i=1}^{r_2-1} d(A^i) - S_2.$$

If $\sum_{i=1}^{r_1-1} d(A^i) = \sum_{i=1}^{r_2-1} d(A^i)$ and $S_1 \leq S_2$, i.e. $r_1 = r_2$ and $k_1 = k_2$, then $t_1 = t_2$.

XV. Cryptographic strength

A. Ciphertext-only attack

Provided that the attacker doesn't have key parameter, i.e. matrix size n and modulus m, then ciphertext-only attack does not seem possible. Matrix size of $n \times n$ modulus m can be presented with m^{n^2} variants.

Suppose we extract a certain ciphertext (r, k, t). Suppose the size of secret matrix is $n \times n$, and operations are performed modulus m. As the matrix is non-degenerate and non-identity, minimum sum of its elements equals $(n+1)$ and maximum $n^2(m-1)$. Thus, initial number lies within the range from $(r-1)(n+1)$ to $rn^2(m-1)$. The sum S lies within the range from 0 to $k(m-1)$ and may be presented with m^k variants.

B. Chosen-plaintext attack

All order-preserving encryptions are vulnerable to such kind of attack. Let us encrypt a sequence of numbers $x_1, x_2, \ldots$. We shall consider ciphertexts of form $(r_i, 0, 0)$. With these values of encryption function, the sum is

$$\sum_{i=1}^{1} d(A^j) = x_i,$$

Hence for subsequent plaintexts $i+1, i+2, \ldots$. We can find residuals

$$\sum_{i=1}^{1} d(A^j) - x_i$$

encrypted with the last two elements of the tuple.

Next we shall consider ciphertexts of form $(r_i, k_t, 0)$. Since the last tuple element equals zero, corresponding residual equals

$$\sum_{j=1}^{k_t} a_j,$$

where a_j are elements of matrix A^{r_1} in accordance with secret traversal. Thus, examining successively $k_1 = 1, k_2 = 2, \ldots$, we can determine the elements of matrix A^r. During security enhancement, matrix elements are summed up randomly, with $n^2!$ possible variants. Besides, if $r > 1$, matrix root is an expensive operation.

Another way to enhance security is applying strictly increasing function value f(x) to initial number x before encryption procedure. For instance, f(x) can be Ax+B, where $A, B \in \mathbb{Z}$, $A > 1$, $0 < B < A$, and instead of x there is initial number. The procedure of encryption substitution is as follows. Suppose, $Num \in \mathbb{Z}_{\geq 0}$ a number, that needs to be encrypted. Let us randomly choose number A, which is the part of the key. Allowed value range for B is limited by A, therefore, the bigger number A, the more possible variants there are for B. Number B is also randomly chosen, and adding number B enables two neighbor numbers to grade into numbers, whose difference is a random number, i.e $Num_2 - Num_1 = A(x+1) + B_2 - (Ax + B_1) = A + B_2 - B_1$.

Thus, in order to find the number, following number Num_2, we need to sort out at worst $A + B_3 - B_2 = 3A - 2$ numbers, which enhances the construct security.

Next we substitute x for Num, calculate Ax+B and encrypt the deduced number. After decryption, we need to perform integer division of decrypted number by A in order to get initial number. This scheme doesn't require storing of coefficient B within the key and, therefore gives opportunity to use different B values for any numbers, including equal numbers.

XV. Statistics

The plots depicting dependence of r, k, t parameters on size of initial unencrypted values are provided below. We used secret matrix of size 10 × 10 with elements from $\mathbb{Z}_{10}$ and parameter $A=2^8$ for encrypting 1000 subsequent 8-bit numbers.

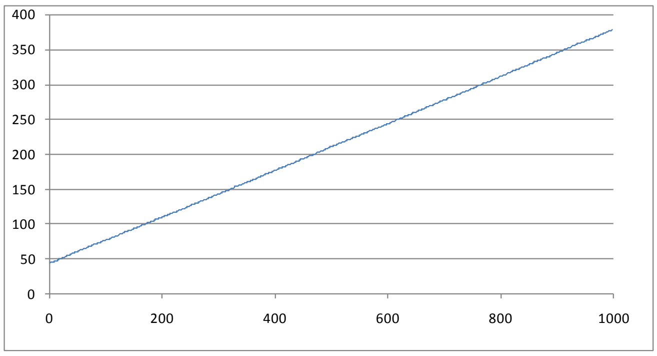

Fig.4 Dependence of r parameter on plaintext size

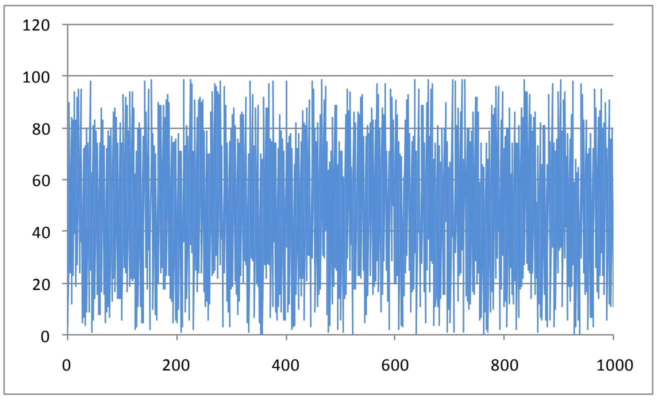

Fig.5 Dependence of k parameter on plaintext size

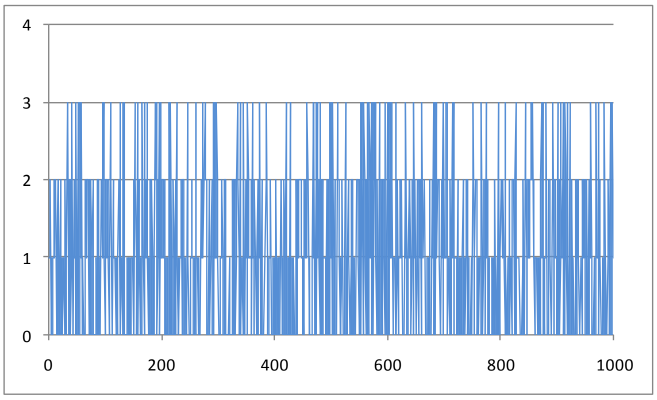

Fig.6 Dependence of t parameter on plaintext size

XVI. Further work

Currently we are examining other ways of attacking given OPE schemes and increasing cryptographic strength of them. We are also looking for optimal parameters of schemes, which provide an acceptable balance of speed and security.

There are several possible approaches to accelerate implementation such as using GPU for calculations, using specified matrix implementation, using faster standard algorithms required in schemes, etc.

After appropriate results are achieved, these schemes will be embedded as third-party libraries in security database service implemented in our laboratory.

References

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

[2] 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, http://dx.doi.org/10.1109/ICDEW.2006.30

[3] R. Agrawal, J. Kiernan, R. Stikant, and Y. Xu, Order-preserving encryption for numeric data, ACM SIGMOD International Conference on Management of Data, pp. 563-574, 2004.

[4] G. Amanatidis, A. Boldyreva, and A. O'Neill, Provably-secure schemes for basic query support in outsourced databases, Working Conference on Data and Applications Security, pp. 14-30, 2007.

[5] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, Order-preserving symmetric encryption, Eurocrypt, pp. 224-241, 2009.

[6] A. Boldyreva, N. Chenette, and A. O'Neill, Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions, Crypto11, 2011.

[7] Schneier, B., Wiley J.: Applied Cryptography Second Edition 1996 ISBN 0-471-11709-9..

[8] D. Boneh and B. Waters, Conjunctive, subset, and range queries on encrypted data, TCC, 535-554, 2007.

[9] H. Hacigumus, B.R. Iyer, C. Li, and S. Mehrotra, Executing SQL over encrypted data in the database-service-provider model, ACM SIGMOD Conference on Management of Data, 2002.

[10] M. Halloush and M. Sharif, Global heuristic search on encrypted data (GHSED), International Journal of Computer Science Issues (IJCSI), 1:13-17, 2009.

[11] Raluca A. Popa, Frank H. Li, Nickolai Zeldovich, An Ideal-Security Protocol for Order-Preserving Encoding. IEEE Symposium on Security and Privacy 2013, http://dx.doi.org/10.1109/CISS.2012.6310814