Key-Controlled Order-Preserving Encryption

HU Mengke[1], Telecom-Sudparis, 91000 Evry, France
Email: Mengke.Hu@telecom-sudparis.eu
GAO Juntao[1], CNIS Laborabory, Xidian University, 710071 Xi'an, China
Email: jtgao@mail.xidian.edu.cn

Abstract. In this paper we study order-preserving encryption (OPE), a primitive in the database community for allowing efficient range queries on ecrypted data. OPE was suggested by Agrawal et al [1], and was throughly studied by Boldyreva et al [2]. In this paper we present a practical OPE scheme, which is a key-controlled algorithm, based on simple computation. A primary analysis shows that our algorithm is secure enough.

Keywords: order-preserving encryption, database security, cryptography

1 Introduction

Order-preserving encryption (OPE) is a peculiar encryption scheme, whose encryption function preserves numerical ordering of plaintexts. Such special feature of OPE makes it useful in database security designing. OPE was suggested by R. Agrawal et al [1], and was throughly studied by Alexandra Boldyreva et al [2]. Alexandra Boldyreva et al [2] proved that OPE can not withstand chosen-plaintext-attack (CPA), so that an OPE machine must be securely holded. Then [2] they proposed an OPE scheme based on Lazy Sampling Algorithm. Finally [2] they proved the security of their scheme, in the security notion "look as random as possible". Their work is excellent, but their scheme is bulky, with large sized Lazy Sampling Algorithm. More important is that, the correctness of their scheme depends on computation precision.

In this paper we present a practical OPE scheme, which is a key-controlled algorithm, based on simple computation. Our scheme is composed of 4 layers, each of which is reversible, and each of which is order-preserving. We can not prove the security of our scheme in the notion "look as random as possible", but a primary analysis shows that our algorithm is secure enough.

2 Computing Components for Constructing OPE

2.1 Notations

For the sake of convenient description, some times we take a bit-string, with the length n, as an integer from the interval $[0, 2^n-1]$. $HW(a_1a_2\cdots a_m)$ is Hamming Weight of bit-string $(a_1a_2\cdots a_m)$ (that is, the number of 1's in the string $(a_0a_1a_2\cdots a_m)$). For an integer a, $\lceil a \rceil$ is the largest integer not larger than a. In this paper the operation "+" is always taken as the addition over real number field, other than "bit-oriented exclusive or".

2.1 Key-Controlled Transformations $f_{(0)}(., k)$ and $f_{(1)}(., k)$

The key $k=(k_0k_1k_2\cdots k_{15})$ is 16 bits long. The input X is 4 bits long. The output Y is 5 bits long. $f_{(0)}(., k)$ is described as

$$Y= f_{(0)}(X, k)=X+HW(k_0k_1k_2\cdots k_X).$$

The computation of inverse $X=f_{(0)}^{-1}(Y, k)$ is easy, which is to find unique value X such that $X+HW(k_0k_1k_2\cdots k_X)=Y$. $f_{(1)}(., k)$ is described as

$$Y= f_{(1)}(X, k)=X+16-HW(k_Xk_{X+1}k_{X+2}\cdots k_{15}).$$

The computation of inverse $X=f_{(1)}^{-1}(Y, k)$ is easy, which is to find unique value X such that $X+16-HW(k_Xk_{X+1}k_{X+2}\cdots k_{15})=Y$.

Both $f_{(0)}(., k)$ and $f_{(1)}(., k)$ are order-preserving functions of the input X. For any key k, any $X^{(1)}$ and $X^{(2)}$ such that $X^{(1)}<X^{(2)}$, we have $f_{(0)}(X^{(1)}, k)<f_{(0)}(X^{(2)}, k)$ and $f_{(1)}(X^{(1)}, k)<f_{(1)}(X^{(2)}, k)$. The difference between $f_{(0)}(., k)$ and $f_{(1)}(., k)$ is that $f_{(0)}(., k)$ tends to be small, while $f_{(1)}(., k)$ tends to be large. We can easily induce Property 1 and Property 2, stated in the follow.

Property 1 For random key k and random input X, the average values of $f_{(0)}(X, k)$ and $f_{(1)}(X, k)$ are respectively 11.75 and 19.25.

Property 2 It always holds that $f_{(0)}(X, k)\leqslant 2X+1$, and it always holds that $f_{(1)}(X, k)\geqslant 2X$.

2.2 Linear expensions $l_{(0)}(., m)$ and $l_{(1)}(., m)$

The input X is n bits long. The output Y is $n+1$ bits long. m is an integer such that $1\leqslant m\leqslant n-1$. $l_{(0)}(., m)$ is described as

$$Y= l_{(0)}(X, m)=X+\left\lceil X/2^{n-m}\right\rceil.$$

The computation of inverse $X=l_{(0)}^{-1}(Y, m)$ is easy. Compute $Y\div(1+2^{n-m})$ to obtain integer quotient u and fraction residual v. Then (1) $w=v\times(1+2^{n-m})$ is an integer such that $0\leqslant w<2^{n-m}$; (2) $X=u\times 2^{n-m}+w$. $l_{(1)}(., m)$ is described as

$$Y= l_{(1)}(X, m)=X+2^n-\left\lceil \left(2^n-1-X\right)/2^{n-m}\right\rceil.$$

The computation of inverse $X=l_{(1)}^{-1}(Y, m)$ is easy. Compute $Z=2^{n+1}-1-Y$, then

$$Z=(2^n-1-X)+\left\lceil \left(2^n-1-X\right)/2^{n-m}\right\rceil.$$

Compute $Z\div(1+2^{n-m})$ to obtain integer quotient u and fraction residual v. Then (1) $w=v\times(1+2^{n-m})$ is an integer such that $0\leqslant w<2^{n-m}$; (2) $(2^n-1-X)=u\times 2^{n-m}+w$.

Both $l_{(0)}(., m)$ and $l_{(1)}(., m)$ are order-preserving functions of the input X. For any parameter m such that $1\leqslant m\leqslant n-1$, any $X^{(1)}$ and $X^{(2)}$ such that $X^{(1)}<X^{(2)}$, we have $l_{(0)}(X^{(1)}, m)<l_{(0)}(X^{(2)}, m)$ and $l_{(1)}(X^{(1)}, m)<l_{(1)}(X^{(2)}, m)$. The difference between $l_{(0)}(., m)$ and $l_{(1)}(., m)$ is that $l_{(0)}(., m)$ tends to be small, while $l_{(1)}(., m)$ tends to be large. We can easily induce that $l_{(0)}(X, m)\leqslant(1+2^{-(n-m)})X$, and that $l_{(1)}(X, m)\geqslant(1+2^{-(n-m)})X+2^n-2^m+2^{-(n-m)}$.

3 Description of Our OPE Scheme

Our scheme encrypts plaintexts with the length 64, into ciphertexts with the length 126. The scheme is composed of 4 layers, with the encryption procedure as:

plaintext→the first layer→the second layer→the third layer→the fourth layer→ciphertext.

3.1 The First Layer

The key $k_{(1)}=\{(k_{(1,1,*)}, k_{(1,1,0)}, k_{(1,1,1)}, \cdots, k_{(1,1,15)}), (k_{(1,2,*)}, k_{(1,2,0)}, k_{(1,2,1)}, \cdots, k_{(1,2,15)}), \cdots, (k_{(1,16,*)}, k_{(1,16,0)}, k_{(1,16,1)}, \cdots, k_{(1,16,15)})\}$ is 272 bits long. The input X is 64 bits long. The output Y is 80 bits long. Denote $X=(X_1, X_2, \cdots, X_{16})$, $Y=(Y_1, Y_2, \cdots, Y_{16})$, where each X_j is 4 bits long, and each Y_j is 5 bits long. Then

$$Y_j = f_{(k_{(1,j,*)})}(X_j,(k_{(1,j,0)},k_{(1,j,1)},\cdots,k_{(1,j,15)})), j=1\sim16.$$

According to subsection 2.1, the first layer is easy to be reversed. It is not difficult to induce that the first layer is an order-preserving function.

3.2 The Second Layer

The key $k_{(2)}=(k_{(2,1)}, k_{(2,2)}, \cdots, k_{(2,20)})$ is 20 bits long. The input X is 80 bits long. The output Y is 100 bits long. This layer is

$$U_1=l_{(k_{(2,1)})}(X,1),$$

$$U_2=l_{(k_{(2,2)})}(U_1,2),$$

$$\cdots$$

$$U_{19}=l_{(k_{(2,19)})}(U_{18},19),$$

$$Y=l_{(k_{(2,19)})}(U_{19},20).$$

According to subsection 2.2, the second layer is easy to be reversed. It is obvious that the second layer is an order-preserving function.

3.3 The Third Layer

The key $k_{(3)}=\{(k_{(3,1,*)}, k_{(3,1,0)}, k_{(3,1,1)}, \cdots, k_{(3,1,15)}), (k_{(3,2,*)}, k_{(3,2,0)}, k_{(3,2,1)}, \cdots, k_{(3,2,15)}), \cdots, (k_{(3,25,*)}, k_{(3,25,0)}, k_{(3,25,1)}, \cdots, k_{(3,25,15)})\}$ is 425 bits long. The input X is 100 bits long. The output Y is 125 bits long. Denote $X=(X_1, X_2, \cdots, X_{25})$, $Y=(Y_1, Y_2, \cdots, Y_{25})$, where each X_j is 4 bits long, and each Y_j is 5 bits long. Then

$$Y_j = f_{(k_{(3,j,*)})}(X_j,(k_{(3,j,0)},k_{(3,j,1)},\cdots,k_{(3,j,15)})), j=1\sim25.$$

According to subsection 2.1, the third layer is easy to be reversed. It is not difficult to induce that the third layer is an order-preserving function.

3.4 The Fourth Layer

Both the key $k_{(4)}=(k_{(4,1)}, k_{(4,2)}, \cdots, k_{(4,125)})$ and the input X are 125 bits long. The output Y is 126 bits long. This layer is $Y=X+k_{(4)}$.

3.5 Key Schedual

We can see that the encryption key is $(k_{(1)}, k_{(2)}, k_{(3)}, k_{(4)})$, with the total length 842. We need to use a key scheduall algorithm, to generate 842 bits of encryption key by a short user key. This is not an important problem, many tools can do this. For example, we can use a simple stream cipher generator.

3.6 Explaining Our OPE Scheme

Each of 5 layers is easy to be reversed, and each of them is order-preserving, so that our scheme is easy to be reversed, and that our scheme is order-preserving.

The function of the second layer is to generate confusion and diffusion. We can see that, in this layer, each bit of the output depends on each bit of the input, and depends on each bit of the key $k_{(2)}$. Because of the restriction “order-preserving”, such confusion and diffusion are not sufficient. But it is enough against those allowed attacks (Notice that the attacker is not allowed to make chosen-plaintext attack. He is only allowed to make known-plaintext attack. He can accumulate random (plaintext, ciphertext) pairs, other than elaborately chosen pairs).

The reason for using the fourth layer is that the output of the third layer is not random enough. Notice that the output of the third layer is

$$Y_j = f_{(k_{(4,j,*)})}(X_j, (k_{(4,j,0)}, k_{(4,j,1)}, \cdots, k_{(4,j,15)})), \, j=1\sim 25.$$

If Y_j tends to be small, the attacker can guess $k_{(4,j,*)}=0$. If Y_j tends to be large, the attacker can guess $k_{(4,j,*)}=1$. The fourth layer covers such non-randomness, so that the fourth layer protects the third layer.

4 Primary Analysis of Our OPE Scheme

4.1 The Strength Against Chosen-Plaintext Attack (CPA)

Although Alexandra Boldyreva et al [2] proved that OPE can not withstand chosen-plaintext-attack (CPA), here we still present an analysis of our OPE scheme against CPA. From our checking, there is only one weak plaintext $X=(0, 0, \cdots, 0)$.

If the plaintext is $X=(0, 0, \cdots, 0)$, the ciphertext Y is a ordinary value, we can not find any useful information about the key. An extreme example is that $X=(0, 0, \cdots, 0)$ and $Y=(0, 0, \cdots, 0)$. In this case we can induce that

$(k_{(1,1,*)}, k_{(1,1,0)})=(0, 0)$ or $(k_{(1,1,*)}, k_{(1,1,0)}, k_{(1,1,1)}, \cdots, k_{(1,1,15)})=(1, 1, 1, \cdots, 1)$;

$(k_{(1,2,*)}, k_{(1,2,0)})=(0, 0)$ or $(k_{(1,2,*)}, k_{(1,2,0)}, k_{(1,2,1)}, \cdots, k_{(1,2,15)})=(1, 1, 1, \cdots, 1)$;

$\cdots$;

$(k_{(1, 16, *)}, k_{(1, 16, 0)})=(0, 0)$ or $(k_{(1, 16, *)}, k_{(1, 16, 0)}, k_{(1, 16, 1)}, \cdots, k_{(1, 16, 15)})=(1, 1, 1, \cdots, 1)$;
$(k_{(2, 1)}, k_{(2, 2)}, \cdots, k_{(2, 20)})=(0, 0, \cdots, 0)$;
$(k_{(3, 1, *)}, k_{(3, 1, 0)})=(0, 0)$ or $(k_{(3, 1, *)}, k_{(3, 1, 0)}, k_{(3, 1, 1)}, \cdots, k_{(3, 1, 15)})=(1, 1, 1, \cdots, 1)$;
$(k_{(3, 2, *)}, k_{(3, 2, 0)})=(0, 0)$ or $(k_{(3, 2, *)}, k_{(3, 2, 0)}, k_{(3, 2, 1)}, \cdots, k_{(3, 2, 15)})=(1, 1, 1, \cdots, 1)$;
$\cdots$;
$(k_{(3, 25, *)}, k_{(3, 25, 0)})=(0, 0)$ or $(k_{(3, 25, *)}, k_{(3, 25, 0)}, k_{(3, 25, 1)}, \cdots, k_{(3, 25, 15)})=(1, 1, 1, \cdots, 1)$;
$(k_{(4, 1)}, k_{(4, 2)}, \cdots, k_{(4, 125)})=(0, 0, \cdots, 0)$.
For any key schedual algorithm, such case is impossible.

5 Conclusion

Our OPE scheme is a key-controlled algorithm, based on simple computation. It is reversible, and is order-preserving. Although we can not prove the security of our scheme in the notion “look as random as possible”, a primary analysis shows that our algorithm is secure enough.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD’04,* pp. 563-574. ACM, 2004.
2. Alexandra Boldyreva, Nathan Chenette, Younho Lee and Adam O’neill. Order-preserving symmetric encryption. In *Advances in Cryptology-EUROCRYPT 2009,* pp. 224-241. Springer, 2009.