Order-Preserving Encryption Using Approximate Integer Common Divisors

James Dyer*[1], Martin Dyer†[2], and Jie Xu‡[2]

[1]School of Computer Science, University of Manchester
[2]School of Computing, University of Leeds

2nd June 2017

Abstract

We present an *order-preserving encryption* (OPE) scheme based on the *general approximate common divisor problem* (GACDP). This scheme only requires $O(1)$ arithmetic operations for encryption and decryption. We have performed extensive evaluation of our algorithms by comparing execution times against those of other OPE schemes.

1 Introduction

Outsourcing computation to the cloud has become increasingly important to business, government, and academia. However, in some circumstances, data on which those computations are performed may be sensitive. Therefore, outsourced computation proves problematic.

To address these problems, we require a means of secure computation in the cloud. One proposal, is that of *homomorphic encryption*, where data is encrypted and computation is performed on the encrypted data. The data is retrieved and decrypted. Because the encryption is homomorphic over the operations performed by the outsourced computation, the decrypted result is the same as that computed on the unencrypted data.

Fully homomorphic encryption has been proposed as a means of achieving this. However, as currently proposed, it is not practical. Therefore, we believe that *somewhat homomorphic encryption*, which is homomorphic only for certain inputs or operations, is only of current practical interest.

For sorting of data, we require an encryption scheme that supports homomorphic comparisons of ciphertexts. *Order-preserving encryption* (OPE) is a

*james.dyer@postgrad.manchester.ac.uk
†m.e.dyer@leeds.ac.uk
‡j.xu@leeds.ac.uk

recent field that supports just such a proposition. An OPE is defined as an encryption scheme where, for plaintexts m_1 and m_2 and corresponding ciphertexts c_1 and c_2,

$$m_1 \leq m_2 \implies c_1 \leq c_2$$

Our work presents an OPE scheme that is based on the *general approximate common divisor problem* (GACDP) [13], which is believed to be hard. Using this problem we have devised a system where encryption and decryption are $O(1)$ arithmetic operations.

1.1 Notation

$x \xleftarrow{\$} S$ represents a value x chosen uniformly at random from the discrete set S.

$\mathsf{KeyGen} : \mathcal{S} \rightarrow \mathcal{K}$ denotes the key generation function operating on the security parameter space $\mathcal{S}$ and whose range is the secret key space $\mathcal{K}$.

$\mathsf{Enc} : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ denotes the symmetric encryption function operating on the plaintext space $\mathcal{M}$ and the secret key space $\mathcal{K}$ and whose range is the ciphertext space $\mathcal{C}$.

$\mathsf{Dec} : \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$ denotes the symmetric decryption function operating on the ciphertext space $\mathcal{C}$ and the secret key space $\mathcal{K}$ and whose range is the plaintext space $\mathcal{M}$.

$m, m_1, m_2, \ldots$ denote plaintext values. Similarly, $c, c_1, c_2, \ldots$ denote ciphertext values.

$[x, y]$ denotes the integers between x and y inclusive.

$[x, y)$ denotes $[x, y] \setminus \{y\}$.

1.2 Scenario

Our OPE system is intended to be employed as part of a system for single-party secure computation in the cloud. In this system, a secure client encrypts data and then outsources computation on the encrypted data to the cloud. Then computation is performed homomorphically on the ciphertexts. The results of the computation are retrieved by the secure client and decrypted. We intend that our OPE scheme will support sorting of encrypted data.

1.3 Formal Model of Scenario

We have n integer inputs, $m_1, m_2, \ldots, m_n$, where $m_i \in \mathcal{M} = [0, M]$. We wish to sort these inputs. A secure client A selects an instance $\mathcal{E}_K$ of the OPE algorithm $\mathcal{E}$ using the secret parameter set K. A encrypts the n inputs by computing $c_i = \mathcal{E}_K(m_i)$, for $i \in [1, n]$. A uploads $c_1, c_2, \ldots, c_n$ to the cloud computing environment. The cloud environment sorts the $c_i, i \in [1, n]$. Since $\mathcal{E}$ is an OPE, the m_i will now be correctly sorted. Then A retrieves some or all of the c_i from the cloud and decrypts a ciphertext c_i by computing $m_i = \mathcal{E}_K^{-1}(c_i)$.

A snooper is only able to inspect $c_1, c_2, \dots, c_n$ in the cloud environment. The snooper may compute additional functions on the $c_1, c_2, \dots, c_n$ as part of a cryptanalytic attack, but cannot make new encryptions.

1.4 Observations from Scenario

From our scenario we observe that we do not require public-key encryption as we do not intend another party to encrypt data. Symmetric encryption will suffice. Furthermore, there is no key escrow or distribution problem as only ciphertexts are distributed to the cloud.

It is common in the literature [3, 2] to refer to an encryption or decryption oracle in formal models of security. However, our scenario has no analogue of an oracle because another party has no way of encrypting or decrypting data without breaking the system. Any cryptological attacks will have to be performed on ciphertexts only. Therefore, we see *chosen plaintext attacks* (CPA) and *chosen ciphertext attacks* (CCA) as not relevant to our scenario. Indeed, it can be argued that any notion of indistinguishability under CPA is not even relevant to OPE in general. Various attempts by Boldyreva and others [5, 4, 25, 27] to provide such a notion have required restricting the model of security so that it is no longer practically relevant. We also note that a *known plaintext attack* (KPA) is possible only by brute force, and not through being given a sample of pairs of plaintext and corresponding ciphertext.

We use a notion of security that guarantees only, that, in the lifetime of the outsourced computation, determining the plaintext values will be computationally infeasible, and any information leaked about the plaintexts as a result of the OPE will not aid an attacker in determining the plaintext values. We consider this notion most relevant to practical implementations.

1.5 Related Work

Prior to Boldyreva et al. [5], OPE had been investigated by Agrawal et al. [1] and others (see [1] for earlier references). However, it wasn't until Boldyreva et al. that it was claimed that an OPE scheme was provably secure. Boldyreva's algorithm constructs a random order-preserving function by mapping M consecutive integers in a domain to integers in a much larger range $[1, N]$ by recursively dividing the range into M monotonically increasing subranges. Each integer is assigned a pseudorandom value in its subrange. The algorithm recursively bisects the range, at each recursion sampling from the domain until it hits the input plaintext value. The algorithm is designed this way because Boldyreva et al. wish to sample uniformly from the range. This would require sampling from the negative hypergeometric distribution, for which no efficient exact algorithm is known. Therefore they sample the domain from the hypergeometric instead. As a result, each encryption requires at least $\log N$ recursions. Furthermore, so that a value can be decrypted, the pseudorandom values generated must be reconstructible. Therefore, for each instance of the algorithm, a plaintext will always encrypt to the same ciphertext. For our OPE scheme, multiple encryptions of a plaintext will produce differing ciphertexts. In [5], the authors claim that $N = 2M$, a claim repeated in [9], although [4] suggests $N \geq 7M$. However both [4, 5] take no account of n, the number of values to be encrypted. These values are assumed to be uniformly distributed in $[1, M]$. If $c = f(m)$ is Boldyreva et al's OPE, we can estimate $f^{-1}(c)$ by Mc/N, with standard deviation approximately $\sqrt{2m(1 - m/M)}$. For this reason, Boldyreva et al.'s scheme always leaks about half the plaintext bits, but the leakage is larger if m is near the extremes of $[1, M]$, This seems quite possible in a large application, where n may be comparable to M. If $m_1 \leq \cdots \leq m_n$, and $M = O(n)$, the scheme will expose m_1 to within a constant number of bits. Since we do not know what n will be, we can only increase uncertainty $O(\sqrt{N/M})$ in c_1. Then M is effectively larger, by comparison with n. If we wish this to be $\Omega(\sqrt{M})$, say, we must take $N = \Omega(M^2)$. Therefore, we use $N \geq M^2$ in our implementations of Boldyreva et al.'s algorithm. This has the additional advantage that their scheme can be approximated closely by a much simplified computation, as we discuss in section 3.2.

Yum et al. [28] extend Boldyreva et al.'s work to non-uniformly distributed plaintexts. This can improve the situation in the event that the client knows the distribution of plaintexts, which seems unlikely, although this "flattening" idea already appears in [1]. However, the benefits can be largely negated if the adversary also knows this distribution.

In [4], Boldyreva et al. suggest an extension to their original scheme, modular order-preserving encryption (MOPE), by simply transforming the plaintext before encryption by adding a term modulo M. The idea is to cope with the problems discussed above, but any added security arises only from this term being unknown. Note also that this construction again always produces the same ciphertext value for each plaintext.

Teranishi et al.[25] devise a new OPE scheme that satisfies their own security model. However, their algorithms are less efficient, being linear in the size of the message space. Furthermore, like Boldyreva et al., a plaintext always encrypts to the same ciphertext value.

Krendelev et al. [17] devise a an OPE scheme based on a coding of an integer as the real number $\sum_i b_i 2^{-i}$ where b_i is the ith bit of the integer. The algorithm to encode the integer is $O(n)$ where n is the number of bits in the integer. Using this encoding, they construct a matrix-based OPE scheme where a plaintext is encrypted as a tuple (r, k, t). Each element of the tuple is the sum of elements from a matrix derived from the private key matrices σ and A. Their algorithms are especially expensive, as they require computation of powers of the matrix A. Furthermore, each plaintext value always encrypts to the same ciphertext value.

Khadem et al. [14] propose a scheme to encrypt equal plaintext values to differing values. Their scheme is similar to Boldyreva et al. where a plaintext is mapped to a pseudorandom value in a subrange. However, this scheme relies on the domain being a set of consecutive integers for decryption. Our scheme easily allows for non-consecutive integers.

Liu et al. [19] addresses frequency of plaintext values by mapping the plaintext value to a value in an extended message space and splitting the message and ciphertext spaces nonlinearly. Like our scheme, decryption is a simple division. However, the ciphertext interval must first be located for a given ciphertext which is $\Omega(\log n)$ where n is the total number of intervals.

Liu and Wang [18] describe a system similar to ours where random "noise" is added to a linear transformation of the plaintext. However, in their examples, the parameters and noise used are real numbers. Unlike our work, the security of such a scheme is unclear.

In [22], Popa et al. discuss an interactive protocol for constructing a binary index of ciphertexts. Although this protocol guarantees ideal security, in that it only reveals the ordering, it is not an OPE. The ciphertexts do not preserve the ordering of the plaintexts, rather the protocol requires a secure client decrypt the ciphertexts, compare the plaintexts, and return only the ordering. It is essentially equivalent to sorting the plaintexts on the secure client and then encrypting them. Popa et al.'s protocol has a high communication cost: $\Omega(n \log n)$. This may be suitable for a database server where the comparisons may be made in a secure processing unit with fast bus communication. However, it is unsuitable for a large scale distributed system where the cost of communication will become prohibitive. [16] improves the communication cost of Popa et al.'s protocol to $\Omega(n)$ but this is still onerous for distributed systems. Kerschbaum further extends this protocol to hide the frequency of plaintexts in [15].

Also of note is *order-revealing encryption* (ORE), a generalisation of OPE introduced by Boneh et al. [6] that only reveals the order of ciphertexts. An ORE is a scheme (C, E, D) where C is a comparator function that takes two ciphertext inputs and outputs '$<$' or '$\geq$', and E and D are encryption and decryption functions. This attempts to replace the secure client's responsibility for plaintext comparisons in Popa's scheme with an exposed function acting on the ciphertexts.

Boneh et al.'s construction uses multilinear maps. However, as stated in Chenette et al. [9], "The main drawback of the Boneh et al. ORE construction is that it relies on complicated tools and strong assumptions on these tools, and as such, is currently impractical to implement".

Chenette et al. offer a more practical construction, with weaker claims to provable security. However, since it encrypts the plaintexts bit-wise, it requires a number of applications of a pseudorandom function f linear in the bit size of the plaintext to encrypt an integer. The security and efficiency of this scheme depends on which pseudorandom function f is chosen.

1.6 Road Map

In section 2, we present our OPE scheme. In section 3, we provide the generic version of Boldyreva et al.'s algorithm and the Beta distribution approximation used in our experiments. In section 4, we discuss the results of experiments on our OPE scheme. Finally, in section 5 we conclude the paper.

2 An OPE scheme using Approximate Common Divisors

Our OPE scheme is the symmetric encryption system (KeyGen, Enc, Dec). The message space, $\mathcal{M}$, is $[0, M]$, and the ciphertext space, $\mathcal{C}$, is $[0, N]$, where $N > M$. We have plaintexts $m_i \in \mathcal{M}, i \in [1, n]$ such that $0 < m_1 \leq m_2 \leq \cdots \leq m_n \leq M$. KeyGen chooses a large enough number k as the secret key, sk, such that $k > M^{\frac{8}{3}}$ (see section 2.4). k does not necessarily need to be prime.

2.1 Encryption

To encrypt $m_i \in \mathcal{M}$, we compute,

$$c_i = \mathsf{Enc}(m_i, \mathsf{sk}) = m_i k + r_i,$$

where $r_i \xleftarrow{\$} (k^{\frac{3}{4}}, k - k^{\frac{3}{4}})$.

2.2 Decryption

To decrypt $c_i \in \mathcal{C}$, we compute,

$$m_i = \mathsf{Dec}(c_i, \mathsf{sk}) = \lfloor c_i/k \rfloor.$$

2.3 Order-preserving property

If $m > m'$, then $c \geq c'$ provided $mk+r > m'k+r'$, if $k(m-m') > (r'-r)$, which follows, since the lhs is at least k, and the rhs is less than $(k-1)$. If $m' = m$, then the order of the encryptions is random, since $\Pr(r' > r) = \frac{1}{2} - 1/k \approx \frac{1}{2}$.

2.4 Security of the Scheme

Security of our scheme is give by the *general approximate common divisor problem* (GACDP), which is believed to be hard. It can be formulated as:

Definition 1 (General approximate common divisor problem.) *Suppose we have n inputs $x_i = pm_i + r_i$, $i \in [1, n]$. We have a bound B such that $|r_i| < B$ for all i. Under what conditions on the variables, m_i and r_i, and the bound B, can an algorithm be found that can uniquely determine p in a time which is polynomial in the total bit length of the numbers involved?*

GACDP is clearly equivalent to breaking our system, since $c_i = m_i k + r_i$. To make the GACDP instances hard, we need $k \gg M$ (see below). Furthermore, we need the m_i to have sufficient entropy to negate a simple "guessing" attack [20]. However, note that the model in [20] assumes that we are able to verify when a guess is correct, which does not seem to be the case here.

Knowing a plaintext, ciphertext pair (m, c) does not allow us to break the system, since $c/m = k + r/m \in [k, k + k/M]$, which is a large interval since

$k \gg M$. A large number of such pairs would give more information, but it still does not seem straightforward to determine k. Thus the system has some resistance to KPA, even though this is excluded by our model of single-party secure computation in the cloud. In particular, we can assume that the effective entropy in the m_i is at least $3 \lg M$, say. This observation allows us to encrypt somewhat smaller plaintexts.

No OPE scheme can avoid leaking some information [5]. Let us consider what our scheme leaks. We have $c = mk + r$, so, if c_1, c_2 are the ciphertexts of m_1, m_2, $c_1 - c_2 = k(m_1 - m_2) + (r_1 - r_2)$, and $(r_1 - r_2)/k \in [-1, 1]$. Hence, if c, c_1, c_2 are the ciphertexts of m, m_1, m_2, we have

$$\frac{m_1 - m}{m_2 - m} \approx \frac{c_1 - c}{c_2 - c}. \tag{1}$$

So the encryption approximately preserves the ratio of differences between plaintexts. Given two plaintext, ciphertext pairs, (m_1, c_1), (m_2, c_2), this could be used to estimate a third plaintext m from its ciphertext c. The error is $(c - c_1)/(c_2 - c_1) + O(1)$, which is $O(1)$ if $c \in [c_1, c_2]$, but could be as large as $\Omega(k)$ if c is far from this range. Furthermore, since we assume that the plaintexts have sufficient entropy to negate guessing, it is difficult to see how such a KPA could be launched.

Howgrave-Graham [13] studied two attacks against GACD, to find divisors d of $a_0 + x_0$ and $b_0 + y_0$, given inputs a_0, b_0 of similar size, with $a_0 < b_0$. The quantities x_0, y_0 are the "offsets". The better attack in [13], $\mathsf{GACD\_L}$, succeeds when $|x_0|, |y_0| < X = b_0^{\beta_0}$, and the divisor $d \geq b_0^{\alpha_0}$ and

$$\beta_0 = 1 - \frac{1}{2}\alpha_0 - \sqrt{1 - \alpha_0 - \frac{1}{2}\alpha_0^2} - \epsilon.$$

where $\epsilon > 0$ is a (small) constant, such that $1/\epsilon$ governs the number of possible divisors which may be output. We will take $\epsilon = 0$. This is the worst case for Howgrave-Graham's algorithm, since there is no bound on the number of divisors which might be output.

Note that $\beta_0 < \alpha_0$, since otherwise $\sqrt{1 - \alpha_0 - \frac{1}{2}\alpha_0^2} \leq 1 - \frac{3}{2}\alpha_0$. This can only be satisfied if $\alpha_0 \leq \frac{2}{3}$. But then squaring both sizes of the inequality implies $\alpha_0 \geq \frac{8}{11} > \frac{2}{3}$, contradicting $\alpha_0 \leq \frac{2}{3}$.

Suppose we take $\alpha_0 = \frac{8}{11}$. Then, to foil this attack, we require $\beta_0 \geq \frac{6}{11}$. For our system we have, $b_0 - a_0 = \max m_i - \min m_i = M$,[1]. To ensure that the common divisor k will not be found we require $b_0^{\alpha_0} \geq k$, so we will take $k = b_0^{8/11}$. Since $b_0 \sim Mk$, this then implies $b_0 = M^{11/3}$. Thus the ciphertexts will then have about $11/3$ times as many bits as the plaintexts. Now $\mathsf{GACD\_L}$ could only succeed for offsets less than $b_0^{\beta_0} = b_0^{6/11} = k^{3/4}$. Thus, we choose our random offsets in the range $(k^{3/4}, k - k^{3/4})$.

Cohn and Heninger [10] give an extension of Howgrave-Graham's algorithm to find the approximate divisor of m integers, where $m > 2$. Unfortunately,

[1] Note this is our M, not Howgrave-Graham's.

their algorithm is exponential in m in the worst case, though they say that it behaves better in practice. On the other hand, [7, Appendix A] claims that Cohn and Heninger's algorithm is worse than brute force in some cases. In our case, the calculations in [10] do not seem to imply better bounds than those derived above.

We note also that the attack of [8] is not relevant to our system, since it requires smaller offsets, of size $O(\sqrt{k})$, than those we use.

For a survey and evaluation of the above and other attacks on GACD, see [12].

2.5 Further Observations

This scheme can be used in conjunction with any other OPE method, i.e. any unknown increasing function $f(m)$ of m. We might consider any integer-valued increasing function, e.g. a polynomial function of m, or Boldyreva et al.'s scheme. If $f(m)$ is this function, then we encrypt m by $c = f(m)k + r$, where $r \xleftarrow{\$} (k^{3/4},\ k - k^{3/4})$, and decrypt by $m = f^{-1}\big(\lfloor c/k \rfloor\big)$. Then 1 becomes

$$\frac{f(m_1) - f(m)}{f(m_2) - f(m)} \approx \frac{c_1 - c}{c_2 - c}. \tag{2}$$

So m_2 is estimated by $m_1 \approx f^{-1}\big(f(m) + (f(m_2) - f(m))(c_1 - c)/(c_2 - c)\big)$. This requires inverting the unknown function f, even if we can break the GACD system, which gives greater security. The disadvantage is that the ciphertext size will increase.

If $f(m)$ is an unknown polynomial function, we solve a polynomial equation to decrypt. The advantage over straight GACD is that, even if we can break the GACD instance, we still have to solve an unknown polynomial equation to break the system. For example, suppose we use the linear polynomial $f(m) = a_1(m+a_0)+s$, where $s \xleftarrow{\$} [0, a_0]$ is random noise. But this gives $c = a_1k(m+a_0)+(ks+r)$, which is our OPE system with a deterministic linear monic polynomial $f(m) \leftarrow m + a_0$, $k \leftarrow a_1k$ and $r \leftarrow ks + r \xleftarrow{\$} [0, a_1k)$, so $f(m)$ contains a single unknown parameter, a_0. More generally, we need only consider monic polynomials, for the same reason.

If $c = f(m)$ is Boldyreva et al's OPE, we can approximately invert f, with error $O(\sqrt{m})$. The advantage of this hybrid scheme over Boldyreva et al.'s is therefore greater security. The advantage over straight GACD is that the estimation error from using (1) increases to $\Omega(\sqrt{m})$, even in the best case. Thus the scheme always has some resistance to KPA.

3 Algorithms of Boldyreva type

In this section we describe generic encryption and decryption algorithms based on Boldyreva et al. which sample from any distribution and which bisect on the domain (section 3.1). We also present an approximation of Boldyreva et al.'s algorithm [5] which samples from the Beta distribution (section 3.2). The approximation and generic algorithms are used in our experimental evaluation presented in section 4.

3.1 Generic Algorithms

Algorithm 1 below constructs a random order-preserving function $f : \mathcal{M} \to \mathcal{C}$, where $\mathcal{M} = [0, M], M = 2^r$, and $\mathcal{C} = [1, N], N \geq 2^{2r}$, so that $c = f(m)$ is the ciphertext for $m \in \mathcal{M}$. Algorithm 1 depends on a pseudorandom number generator, P, and a deterministic seed function, S. Likewise, Algorithm 2 constructs the inverse function $f^{-1} : \mathcal{C} \to \mathcal{M}$ so that $m = f^{-1}(c)$.

Algorithm 1 Generic Boldyreva-type Encryption Algorithm

---

function RecursiveEncrypt$(a, b, f(a), f(b), m)$
    $x \leftarrow (a + b)/2$
    $y \leftarrow f(b) - f(a)$
    Initiate P with seed $S(a, b, f(a), f(b))$
    Determine $z \in [0, y]$ pseudorandomly, so that $\Pr(z \notin [y/4, 3y/4])$ is negligible
    $\triangleright$ The condition implies that y cannot become smaller than $3N/4(1/4)^r = 3N/4M^2 = 3M/4$, with high probability.
    $f(x) \leftarrow f(a) + z$
    if $x = m$ **then**
        return $f(x)$
    else if $x > m$ **then**
        return RecursiveEncrypt$(a, x, f(a), f(x), m))$
    else
        return RecursiveEncrypt$(x, b, f(x), f(b), m)$
    end if
end function
Initiate P with a fixed seed S_0.
Choose $f(0), f(M)$ pseudorandomly so that $f(M) - f(0) > 3N/4$
return RecursiveEncrypt$(0, M, f(0), f(M), m)$

3.2 An Approximation

We have a plaintext space,$[1, M]$, and ciphertext space, $[1, N]$. Boldyreva et al. use bijection between strictly increasing functions $[1, M] \to [1, N]$ and subsets of size M from $[1, N]$, so there are $\binom{N}{M}$ such functions. There is a similar bijection between nondecreasing functions $[1, M] \to [1, N]$ and multisets of size M from $[1, N]$, and there are $N^M/M!$ such functions. If we sample n points from such a function f at random, the probability that $f(m_1) = f(m_2)$ for any $m_1 \neq m_2$ is at most $\binom{n}{2} \times 1/N < n^2/2N$. We will assume that $n \ll \sqrt{N}$, so $n^2/2N$ is negligible. Hence we can use sampling either with or without replacement, whichever is more convenient.

Algorithm 2 Generic Boldyreva-type Decryption Algorithm

function RecursiveDecrypt$(a, b, f(a), f(b), c)$
    $x \leftarrow (a+b)/2$
    $y \leftarrow f(b) - f(a)$
    Initiate P with seed $S(a, b, f(a), f(b))$
    Determine $z \in [0, y]$ pseudorandomly
    $f(x) \leftarrow f(a) + z$
    if $f(x) = c$ **then**
        return x
    else if $f(x) > c$ **then**
        return RecursiveDecrypt$(a, x, f(a), f(x), c)$
    else
        return RecursiveDecrypt$(x, b, f(x), f(b), c)$
    end if
end function
Initiate P with a fixed seed S_0.
Choose $f(0), f(M)$ pseudorandomly so that $f(M) - f(0) > 3N/4$
return RecursiveDecrypt$(0, M, f(0), f(M), c)$

Suppose we have sampled such a function f at points $m_1 < m_2 < \cdots < m_k$, and we now wish to sample f at m, where $m_i < m < m_{i+1}$. We know $f(m_i) = c_i$, $f(m_{i+1}) = c_{i+1}$, and let $f(m) = c$, so $c_i \leq c \leq c_{i+1}$.[2] Let $x = m - m_i$, $a = m_{i+1} - m_i - 1$, $y = c - c_i$, $b = c_{i+1} - c_i + 1$, so $1 \leq x \leq a$ and $0 \leq y \leq b$. Write $\tilde{f}(x) = f(x + m_i) - c_i$. Then, if we sample a values from $[0, b]$ independently and uniformly at random, $c - c_i$ will be the xth smallest. Hence we may calculate, for $0 \leq y \leq b$,

$$\Pr\left(\tilde{f}(x) = y\right) = \frac{a!}{(x-1)!\,(a-x)!} \left(\frac{y}{b}\right)^{x-1} \frac{1}{b} \left(\frac{b-y}{b}\right)^{a-x} \tag{3}$$

This is the probability that we sample one value y, $(x-1)$ values in $[0, y)$ and $(a-x)$ values in $(y, b]$, in any order. If b is large, let $z = y/b$, and $\mathrm{d}z = 1/b$, then (3) is approximated by a continuous distribution with, for $0 \leq z \leq 1$,

$$\Pr\left(z \leq \tilde{f}(x)/b < z + \mathrm{d}z\right) = \frac{z^{x-1}(1-z)^{a-x}}{\mathrm{B}(x, a-x+1)}\,\mathrm{d}z \tag{4}$$

which is the $\mathrm{B}(x, a-x+1)$ distribution. Thus we can determine $f(m)$ by sampling from the Beta distribution to $\lg N$ bits of precision. In fact, we only need $\lg b$ bits. However, using $n \leq M \leq \sqrt{N}$,

$$\Pr(\exists i : m_{i+1} - m_i < N^{1/3}) \;\leq\; n\frac{N^{1/3}}{N} \;\leq\; \frac{M}{N^{2/3}} \;\leq\; \frac{1}{N^{1/6}}$$

[2] We can have equality because we sample with replacement.

is very small, so we will almost always need at least $\frac{1}{3} \lg N$ bits of precision. Thus the approximation given by (4) remains good even when $a = 1$, since it is then the uniform distribution on $[0, b]$, where $b \geq N^{1/3}$ with high probability.

When the m_i arrive in random order, the problem is to encrypt them consistently without storing and sorting them. Boldyreva does this using binary search. It might be more efficient to use Fibonacci or golden section search, but subdividing the intervals is easier with binary search. If $M = 2^r$, we will always have $a = 2^s$ and $x = 2^{s-1}$ in (4), so $a - x = x$, and (4) simplifies to

$$\Pr\left(z \leq \tilde{f}(x)/b < z + \mathrm{d}z\right) = \frac{z^{x-1}(1-z)^x}{\mathrm{B}(x, x+1)}\,\mathrm{d}z,$$

for $0 \leq z \leq 1$, This might be closely approximated by a Normal distribution if Beta sampling is too slow.

To make sure that the values are consistent, we use a pseudorandom number generator with a seed which is always the same when generating the search value $c = f(m)$, where m subdivides the interval $[l, r] \subseteq [1, M]$. One way of doing this might be to have a fixed seed, and XOR it with some hash of (a, b). See below for some further details. The search will require $\lg M$ Beta samples, but the values for first few levels of the search could be cached in order to speed up the encryption. Note that Boldyreva's search requires $\lg N$ samples in the worst case, since the branching is done on the values of f rather than m. Decryption can be carried out using the same search procedure, with similar efficiency.

4 Experimental Results

To evaluate our scheme, we conducted a simple experiment to pseudorandomly generate and encrypt 10,000 ρ-bit integers. The ciphertexts were then sorted using a customised TeraSort MapReduce (MR) algorithm [21]. Finally, the sorted ciphertexts were decrypted and it was verified that the plaintexts were also correctly sorted. The MR algorithm was executed on a Hadoop cluster of one master node and 16 slaves. Each node was a Linux virtual machine (VM) having 1 vCPU and 2GB RAM. The VMs were hosted in a heterogeneous OpenNebula cloud. In addition, a secure Linux VM having 2 vCPUs and 8 GB RAM was used to generate/encrypt and decrypt/verify the data.

Our implementation is pure, unoptimised Java utilising the JScience library [11] arbitrary precision integer classes. It is denoted as algorithm *GACD* in Table 1. In addition, to provide comparison for our algorithm we have implemented Boldyreva et al.'s algorithm [5] along with two variants of the Boldyreva et al. algorithm. These latter variants are based on our generic version of Boldyreva et al.'s algorithm (see section 3.1). One is an approximation of Boldyreva'a algorithm which samples ciphertext values from the Beta distribution (referred to as *Beta* in Table 1). The derivation of this approximation is given in section 3.2. The second samples ciphertexts from the uniform distribution (referred to as *Uniform* in Table 1). This variant appears in Popa et al.'s CryptDB [23] source code [24] as `ope-exp.cc`. The mean timings for each experimental configuration

Algorithm	ρ	Encryption		MR Job	Decryption	
		Init. (ms)	Enc. (μs)	Exec. (s)	Init. (ms)	Dec. (μs)
GACD	7	50.13	1.51	63.79	11.62	1.47
GACD	15	58.04	2.18	61.28	10.86	2.46
GACD	31	58.66	2.07	63.02	12.18	2.59
GACD	63	70.85	1.94	65.20	10.61	4.22
GACD	127	91.94	2.38	61.08	11.10	6.29
Boldyreva	7	143.72	191.48	70.78	154.01	192.42
Boldyreva	15	135.04	74390.95	65.47	148.29	79255.23
Beta	7	189.52	57.87	64.77	208.16	58.27
Beta	15	202.64	124.79	63.70	218.91	121.53
Beta	31	181.14	221.92	63.64	208.22	221.83
Beta	63	176.24	477.23	66.74	193.03	466.03
Uniform	7	167.66	42.61	64.64	182.27	42.92
Uniform	15	166.98	83.40	66.29	176.14	82.53
Uniform	31	162.11	179.92	63.89	176.53	180.52
Uniform	63	156.53	409.13	63.91	173.57	412.79
Uniform	127	162.17	1237.34	65.30	170.74	1232.19

Table 1: Timings for each experimental configuration. ρ denotes the bit length of the unencrypted inputs. *Init* is the initialisation time for the encryption/decryption algorithm, *Enc* is the mean time to encrypt a single integer, *Exec* is the MR job execution time, *Dec* is the mean time to decrypt a single integer.

is tabulated in Table 1. The chosen values of ρ for each experimental configuration are as a result of the implementations of Boldyreva et al. and the Beta distribution version of the generic Boldyreva algorithm. The Apache Commons Math [26] implementations of the hypergeometric and Beta distributions we used only support Java signed integer and signed double precision floating point parameters respectively, which account for the configurations seen in Table 1. To provide fair comparison, we have used similar configurations throughout. It should be pointed out that, for the *Boldyreva*, *Beta* and *Uniform* algorithms, when $\rho = 7$, this will result in only 128 possible ciphertexts, even though we have 10,000 inputs. This is because these algorithms will only encrypt each plaintext to a unique value. Such a limited ciphertext space makes these algorithms trivial to attack. Our algorithm will produce 10,000 different ciphertexts as a result of the "noise" term. Each ciphertext will have an effective entropy of at least 21 bits for $\rho = 7$ (see section 2.4). So, our algorithm is more secure than those compared for low entropy inputs.

As shown by Table 1, our work compares very favourably with the other schemes. The encryption times of our algorithm outperform the next best algorithm (*Uniform*) by factors of 28 ($\rho = 7$) to 520 ($\rho = 127$). Furthermore, the decryption times grow sublinearly in the bit length of the inputs. Compare this with the encryption and decryption times for the generic Boldyreva algorithms which, as expected, grow linearly in the bit length of the inputs. Boldyreva's version performs even worse. We believe this is down to the design of the algorithm which executes n recursions where n is the size of the ciphertext space in bits. We also discovered that the termination conditions of their algorithm can result in more recursions than necessary.

It should also be noted that the size of the ciphertext generated by each algorithm seems to have minimal bearing on the MR job execution time. Table 1 shows that the job timings are similar regardless of algorithm.

Of course, it is impossible to compare the security of these systems experimentally, since this would involve simulating unknown attacks. But we believe strongly that the GACD approach gives a better theoretical guarantee of security than do criteria like those of [4, 5], that are not based on the conjectured hardness of any known computational problem.

5 Conclusion

Our work has produced an OPE scheme based on the general approximate common divisor problem (GACDP). We have discussed the scheme in section 2 and conducted experiments to evaluate its efficacy which are reported in section 4. Our results show that our scheme is very efficient with $O(1)$ arithmetic operations for encryption and decryption. However, As a trade-off for the time complexity of our algorithms, our scheme produces large ciphertexts, ~ 3.67 times the number of bits of the plaintext. However, as pointed out in the previous section (section 4), ciphertext sizes had minimal impact on the running time of our MR job used in our experiments.

It should be noted that like any "true" OPE, our scheme cannot guarantee any indistinguishability under CPA [5] unlike the non-OPE protocols of Popa and others [22, 16]. However, we believe it is strong enough for the purpose for which it is intended: outsourcing of computation to the cloud.

Although we have not devised this scheme with databases in mind, we note that, unlike similar schemes, we do not require the message space to be consecutive integers. This means that we can support updates without worrying about overlapping "buckets" as in other schemes.

With regard to our stated purpose, we believe that the experimental results show that the efficiency of our scheme make it suitable for computation in the cloud.

References

[1] R. Agrawal et al. 'Order Preserving Encryption for Numeric Data'. In: *Proc. SIGMOD 2004*. ACM, 2004, pp. 563–574. DOI: `10.1145/1007568.1007632`.

[2] M. Bellare et al. 'A Concrete Security Treatment of Symmetric Encryption'. In: *Proc. FOCS 1997*. 1997, pp. 394–403. DOI: `10.1109/SFCS.1997.646128`.

[3] M. Bellare et al. 'Relations Among Notions of Security for Public-Key Encryption Schemes'. In: *Proc. CRYPTO 1998*. Springer, 1998, pp. 26–45. DOI: `10.1007/BFb0055718`.

[4] A. Boldyreva et al. 'Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions'. In: *Proc. CRYPTO 2011*. Springer, 2011, pp. 578–595. DOI: `10.1007/978-3-642-22792-9_33`.

[5] A. Boldyreva et al. 'Order-Preserving Symmetric Encryption'. In: *Proc. EUROCRYPT 2009*. Springer, Apr. 2009, pp. 224–241. ISBN: 978-3-642-01001-9. DOI: `10.1007/978-3-642-01001-9_13`.

[6] D. Boneh et al. 'Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation'. In: *Proc. EUROCRYPT 2015, Part II*. Springer, 2015, pp. 563–594. DOI: `10.1007/978-3-662-46803-6_19`.

[7] Y. Chen and P. Q. Nguyen. *Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers*. Cryptology ePrint Archive, Report 2011/436. `http://eprint.iacr.org/2011/436`. 2011.

[8] Y. Chen and P. Q. Nguyen. 'Faster Algorithms for Approximate Common Divisors: Breaking Fully Homomorphic Encryption Challenges over the Integers'. In: *Proc. EUROCRYPT 2012*. Springer, 2012, pp. 502–519. DOI: `10.1007/978-3-642-29011-4_30`.

[9] N. Chenette et al. 'Practical Order-Revealing Encryption with Limited Leakage'. In: *Proc. FSE 2016, Revised Selected Papers*. Springer, 2016, pp. 474–493. ISBN: 978-3-662-52993-5. DOI: `10.1007/978-3-662-52993-5_24`.

[10] H. Cohn and N. Heninger. 'Approximate common divisors via lattices'. In: *Proc. ANTS-X*. Vol. 1. Mathematical Sciences Publishers, 2012, pp. 271–293. DOI: 10.2140/obs.2013.1.271.

[11] J.-M. Dautelle. *JScience*. Version 4.3.1. Sept. 2014. URL: http://jscience.org.

[12] S. D. Galbraith et al. 'Algorithms for the approximate common divisor problem'. In: *LMS Journal of Computation and Mathematics* 19.A (2016), pp. 58–72. DOI: 10.1112/S1461157016000218.

[13] N. Howgrave-Graham. 'Approximate Integer Common Divisors'. In: *Cryptography and Lattices*. Springer, 2001, pp. 51–66. DOI: 10.1007/3-540-44670-2_6.

[14] H. Kadhem et al. 'MV-OPES: Multivalued-Order Preserving Encryption Scheme: A Novel Scheme for Encrypting Integer Value to Many Different Values'. In: *IEICE Transactions on Information and Systems* 93.9 (2010), pp. 2520–2533. DOI: 10.1587/transinf.E93.D.2520.

[15] F. Kerschbaum. 'Frequency-Hiding Order-Preserving Encryption'. In: *Proc. CCS 2015*. ACM, 2015, pp. 656–667. DOI: 10.1145/2810103.2813629.

[16] F. Kerschbaum and A. Schroepfer. 'Optimal Average-Complexity Ideal-Security Order-Preserving Encryption'. In: *Proc. CCS 2014*. ACM, 2014, pp. 275–286. DOI: 10.1145/2660267.2660277.

[17] S. F. Krendelev et al. 'Order-preserving encryption schemes based on arithmetic coding and matrices'. In: *Proc. FedCSIS 2014*. 2014, pp. 891–899. DOI: 10.15439/2014F186.

[18] D. Liu and S. Wang. 'Programmable Order-Preserving Secure Index for Encrypted Database Query'. In: *Proc. CLOUD 2012*. 2012, pp. 502–509. DOI: 10.1109/CLOUD.2012.65.

[19] Z. Liu et al. 'New order preserving encryption model for outsourced databases in cloud environments'. In: *Journal of Network and Computer Applications* 59 (2016), pp. 198 –207. DOI: 10.1016/j.jnca.2014.07.001.

[20] J. L. Massey. 'Guessing and Entropy'. In: *Proc. ISIT 1994*. 1994, p. 204. DOI: 10.1109/ISIT.1994.394764.

[21] O. O'Malley. *TeraByte Sort on Apache Hadoop*. Tech. rep. Yahoo, Inc., May 2008, pp. 1–3. URL: http://sortbenchmark.org/YahooHadoop.pdf.

[22] R. A. Popa et al. 'An Ideal-Security Protocol for Order-Preserving Encoding'. In: *Proc. SP 2013*. May 2013, pp. 463–477. DOI: 10.1109/SP.2013.38.

[23] R. A. Popa et al. 'CryptDB: Protecting Confidentiality with Encrypted Query Processing'. In: *Proc. SOSP 2011*. ACM, 2011, pp. 85–100. DOI: 10.1145/2043556.2043566.

[24] R. A. Popa et al. *CryptDB*. Mar. 2014. URL: https://css.csail.mit.edu/cryptdb/.

[25] I. Teranishi et al. 'Order-Preserving Encryption Secure Beyond One-Wayness'. In: *Proc. ASIACRYPT 2014*. Springer, 2014, pp. 42–61. DOI: 10.1007/978-3-662-45608-8_3.

[26] The Apache Software Foundation. *Commons Math: The Apache Commons Mathematics Library*. Version 3.6.1. Aug. 2016. URL: http://commons.apache.org/proper/commons-mat

[27] L. Xiao and I.-L. Yen. 'Security analysis for order preserving encryption schemes'. In: *Proc. CISS 2012*. 2012, pp. 1–6. DOI: 10.1109/CISS.2012.6310814.

[28] D. H. Yum et al. 'Order-Preserving Encryption for Non-uniformly Distributed Plaintexts'. In: *Proc. WISA 2011, Revised Selected Papers*. Springer, 2012, pp. 84–97. DOI: 10.1007/978-3-642-27890-7_7.