Article:
Dyer, J orcid.org/0000-0001-5811-5263, Dyer, M and Djemame, K (2019) Order-Preserving Encryption Using Approximate Common Divisors. Journal of Information Security and Applications, 49. ARTN: 102391. ISSN 2214-2126

https://doi.org/10.1016/j.jisa.2019.102391

Order-Preserving Encryption Using Approximate Common Divisors*

James Dyer[a], Martin Dyer[b], Karim Djemame[b]

[a]*Department of Computer Science, University of Huddersfield, Huddersfield, HD1 3DH, UK.*
[b]*School of Computing, University of Leeds, Leeds, LS2 9JT, UK.*

Abstract

Order-preservation is a highly desirable property for encrypted databases as it allows range queries over ciphertexts. *Order-preserving encryption* (OPE) is used in the encrypted database systems CryptDB and Cipherbase. The former has been adopted by several commercial organisations and the latter was developed as an extension of Microsoft's SQLServer. We present two novel, but simple, randomised OPE schemes based on the *general approximate common divisor problem* (GACDP) and *decisional polynomial approximate common divisor problem* (DPolyACDP) respectively. These appear to be the first OPE schemes to be based on a computational hardness primitive, rather than a security game. Our GACDP based scheme is very efficient, requiring only $O(1)$ arithmetic operations for encryption and decryption. Our DPolyACDP based scheme is similarly efficient. We show that these schemes have near optimal information leakage. We demonstrate how our OPE schemes can be integrated into a secure distributed computing system which computes over encrypted data. We report on an extensive evaluation of our GACDP-based algorithms in such a scenario, a MapReduce computation over encrypted data. The results clearly demonstrate extremely favourable execution times in comparison with existing OPE schemes.

Keywords: order-preserving encryption, secure distributed computing, symmetric cipher, approximate common divisors

1. Introduction

Outsourcing computation has become increasingly important to business, government, and academia. This computation is typically performed in distributed computing platforms such as clouds, grids, or high-performance computing

*A preliminary version [30] of this paper was presented at DPM 2017. This work is supported in part by the European Commission under H2020-ICT-20152 contract 687584 – Transparent heterogeneous hardware Architecture deployment for eNergy Gain in Operation (TANGO) project [26] – and by a Microsoft Azure for Research sponsorship.

Email addresses: `j.e.dyer2@hud.ac.uk` (James Dyer), `m.e.dyer@leeds.ac.uk` (Martin Dyer), `k.djemame@leeds.ac.uk` (Karim Djemame)

(HPC) clusters. 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 these platforms. While cryptography can ensure privacy of data at rest or in transit, commonly used ciphers, such as Advanced Encryption Standard (AES) [52], do not allow computations to be meaningfully performed on ciphertexts. Therefore, if our data was encrypted using such a cipher, it would have to be decrypted before it could be computed upon. Obviously, once decrypted, the data is then exposed. One proposal to this problem is hardware to allow secure computation, such as Intel's Software Guard Extensions (SGX) [33]. Using this hardware, data can be decrypted and computed upon in a secure area of memory or a secure processor. However, as it is hardware dependent, this may not be suitable for some computing environments, particularly heterogeneous computing platforms. Another proposal is that of secure multiparty computation (MPC) protocols. These protocols allow multiple parties to engage in computation of a function without gaining knowledge of any other party's inputs. Great progress has been made on practical implementations of MPC [42]. However, work on scalable MPC has been slower, with many MPC schemes suffering from poor communication and computation costs [60]. Practical scalable MPC protocols have relied on specific hardware [7, 49]. Also proposed, is *homomorphic encryption*, where data is encrypted and computation is performed on the encrypted data [59]. 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. This scheme has advantages in that it is not hardware dependent, making it suitable for heteregeneous distributed systems, and that the party responsible for computation only has access to ciphertexts.

Fully homomorphic encryption has been proposed as a means of achieving this. However, as currently proposed, it is not practical. FHE schemes compute over arithmetic circuits [6] which are a space inefficient representation of computation. In addition, the computation at each gate of a cicruit is performed on encryptions of bits and the ciphertexts are typically large. Implementations of FHE have been considerably slower than computation on plaintexts [3, 27, 65]. Therefore, we believe that *somewhat homomorphic encryption*, which is homomorphic only for certain inputs or operations, is only of current practical interest.

For sorting and comparison of data we require an encryption scheme that supports homomorphic comparisons of ciphertexts. *Order-preserving encryption* (OPE) is a 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,[1]

$$m_1 < m_2 \implies c_1 < c_2$$

[1]This relationship is typically represented as $m_1 \leq m_2 \implies c_1 \leq c_2$. However, this seems to introduce an insecurity, by permitting an equality test for plaintexts using two comparisons.

Order-preserving is a highly desirable property for encrypted databases as it allows range queries over ciphertexts. OPE is used in CryptDB [55] and Cipherbase [5]. CryptDB has been adopted by several commercial organisations [57] and Cipherbase was developed as an extension of Microsoft's SQLServer. Additionally, OPE was investigated by Kerschbaum et al. [37, 38] for integration in SAP.

Our contribution is two novel OPE schemes whose proof of security is based on a computational hardness assumption rather than a security game. Our first scheme is based on the *general approximate common divisor problem* (GACDP) [34]. We have generalised this scheme to n-vectors. Our second scheme is based on the related *decisional polynomial approximate common divisor problem* (DPolyACDP) [20]. We believe that these are the first OPE systems whose underlying security is based on a computationally hard problem. Furthermore, our schemes are very efficient. The GACDP-based system only requires $O(1)$ arithmetic operations for encryption and decryption. Our vector and polynomial based schemes are similarly efficient. This computational efficiency makes our schemes ideally suited for the application context outlined in section 2.

In section 2 of this paper, we describe our usage scenario. In section 3 we discuss related work. In section 4, we present our OPE scheme and a vector-based variant. In section 5, we describe an alternate variant based on DPolyACDP. In section 6, we provide the generic version of Boldyreva et al.'s algorithm and the Beta distribution approximation used in our experiments. In section 7, we discuss various leakage abuse attacks on OPE, particularly with reference to our own schemes. In section 8, we discuss the results of experiments on our GACD-based OPE scheme. Finally, in section 9 we conclude the paper.

2. Background

2.1. Notation

The following notation is used throughout this paper:

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

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

$\mathsf{Enc} : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ 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 $\mathcal{C}$.

$\mathsf{Dec} : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$ 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 $\mathcal{M}$.

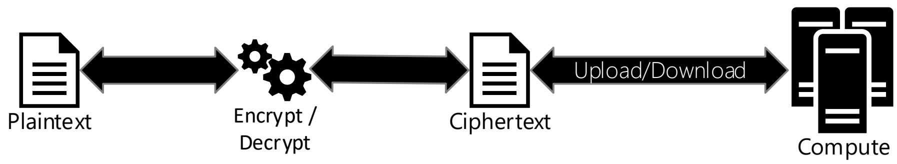
$m, m_1, m_2, \ldots$ denote plaintext values. Similarly, $c, c_1, c_2, \ldots$ denote ciphertext values.

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

(x, y) denotes the integers between x and y exclusive.

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

Figure 1: Scenario



$\mathbb{R}[x, y)$ denotes the real numbers in the interval $[x, y)$.
$\mathbb{Z}[x]$ denotes the set of polynomials with integer coefficients.
$\vec{v}$ denotes a vector.
$\mathbf{r}$ denotes a random variable.

2.2. Scenario

Our OPE system is intended to be employed as part of a system for single-party secure computation in the outsourced distributed computing environment. In this system, a secure client encrypts data and then outsources computation on the encrypted data to the distributed computing environment. Then computation is performed homomorphically on the ciphertexts (see Figure 1). The results of the computation are retrieved by the secure client and decrypted. We intend that our OPE scheme will support sorting and comparison of encrypted data.

2.3. Formal Model of Scenario

We have n integer inputs, $m_1, m_2, \ldots, m_n$, where $m_i \in \mathcal{M} = [0, M]$ and $n \ll M$.

We wish to be able to compare and sort the inputs. A secure client A selects an instance $\mathsf{Enc}(K, \cdot)$ of the OPE algorithm Enc using the secret parameter set K. A encrypts the n inputs by computing $c_i = \mathsf{Enc}(K, m_i)$, for $i \in [1, n]$. A uploads $c_1, c_2, \ldots, c_n$ to the distributed computing environment. These encryptions do not all need to be uploaded at the same time but n is a bound on the total number of inputs. The computing environment conducts comparisons on the $c_i, i \in [1, n]$. Since Enc is an OPE, the m_i will also be correctly sorted. A can retrieve some or all of the c_i from the computing platform and decrypt each ciphertext c_i by computing $m_i = \mathsf{Dec}(K, c_i)$.

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

2.4. 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 computing environment.

Suppose that an attacker interactively submits plaintexts to the data owner to be encrypted so that they are able to view the ciphertexts stored in the cloud. This would make the data owner an encryption oracle [8, 9]. However, in our scenario, the source data set is static, and an attacker is not able to interactively submit plaintexts to the data owner. Furthermore, even if an attacker submits a plaintext to the data owner for inclusion in the source data set, no data is uploaded to the cloud that is not used in the computation. Additionally, no data is uploaded unencrypted. This prevents an attacker linking submitted plaintexts to their encryptions in the cloud. Additionally, since the number of plaintexts is much smaller than the size of the plaintext space, an attacker cannot use the ordering on the ciphertexts to determine which ciphertext corresponds to his submitted ciphertext because they do not have knowledge of the ordering of the plaintexts. Therefore, *chosen plaintext attacks* (CPA) are infeasible in this scenario. Similarly, as the data owner is the only party that can view the decrypted data, *chosen ciphertext attacks* (CCA) are also infeasible and there is no analogue of a decryption oracle. Furthermore, fields that are of low entropy are not encrypted using OPE to prevent frequency attacks (see section 7. Any cryptological attacks will have to be performed on ciphertexts only.

We also note that, for the reasons given above, a *known plaintext attack* (KPA) is infeasible as an attacker is unable to link known plaintexts to corresponding ciphertext.

3. Related Work

Prior to Boldyreva et al. [13], OPE had been investigated by Agrawal et al. [2] and others (see [2] for earlier references). However, it wasn't until Boldyreva et al. that it was claimed that an OPE scheme was provably secure. Boldyreva et al.'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. This implies that the encryption of low entropy data might be very easy to break by a "guessing" attack (see section 8). For our OPE scheme, multiple encryptions of a plaintext will produce differing ciphertexts. In [13], the authors claim that $N = 2M$, a claim repeated in [18], although [14] suggests $N \geq 7M$. We use $N \geq M^2$ in our implementations of Boldyreva et al.'s algorithm, since this has the advantage that the scheme can be approximated closely by a much simplified computation, as we discuss in section 6.2. The cost is only a doubling of the ciphertext size. However both [13, 14] take no account of n, the number of values to be encrypted. As in our scheme, the scheme should have $n \ll M$ to avoid the sorting attack of [51]. If $c = f(m)$ is Boldyreva et al.'s OPE, it is straightforward to show that we can estimate $f^{-1}(c)$ by $\hat{m} = Mc/N$, with standard deviation approximately $\sqrt{2\hat{m}(1 - \hat{m}/M)}$. For this reason, Boldyreva et al.'s scheme always leaks about half the plaintext bits.

Yum et al. [69] 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. This "flattening" idea already appears in [2]. In 7.2 we discuss a similar idea.

In [14], 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 some of the problems discussed above, but any additional 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.[63] 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. [41] 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. [35] 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 allows for non-consecutive integers. This means that our scheme can support updates without worrying about overlapping "buckets" as Khadem et al.

Liu et al. [47] 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. As in our scheme, decryption is a simple division. However, the ciphertext interval must first be located for a given ciphertext which is $\Omega(\log n)$ when n is the total number of intervals.

Liu and Wang [46] 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.

Khoury et al. [39] describe an OPE scheme where the ciphertext is an integer multiple of the logarithm of the plaintext. The security of such a scheme is unclear as no security analysis is provided.

In [56], Popa et al. discuss a stateful interactive protocol for constructing a binary index of ciphertexts. Although this protocol guarantees ideal security, $O(n \log n)$ bit leakage, 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 to decrypt the ciphertexts, compare the plaintexts, and return 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. Kerschbaum and Schroepfer [38] improved the communication cost of Popa et al.'s protocol to $\Omega(n)$ under the assumption that the input is uniformly distributed. However, this is still onerous for distributed systems. Kerschbaum [37] further extends this protocol to hide the frequency of plaintexts. Boelter et al.[12] extend Popa et al.'s idea by using "garbled circuits" to obfuscate comparisons. However, the circuits can only be used once, so their system is one-time use.

In [58], Quan et al. describe a stateful deterministic mutable OPE scheme that supports top-k queries while minimising bit leakage from ciphertexts not in the top-k. Kim et al. [40] also describe a stateful deterministic OPE using a similar methodology to Boldyreva et al. [13]. Here, a ciphertext is composed of two parts: an order-preserving encoding and an encryption using a symmetric key cipher.

Yang et al [68] detail a semi-order preserving scheme. In this scheme, multiple plaintexts may be mapped to the same ciphertext, so a state is maintained to allow decryption.

Also of note is *order-revealing encryption* (ORE), a generalisation of OPE introduced by Boneh et al. [15], 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. [18], "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". In addition, recent work [22] on recovering the secret parameters of CLT2013 has indicated that encryption schemes based on multilinear maps may not be secure.

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.

Lewi et al. [44] devise an ORE scheme where there are two modes of encryption: left and right. The left encryption consists of a permutation of the domain and a key generated by hashing the permuted plaintext value. The right ciphertext consists of encryptions of the comparison with every other value in the domain. It is a tuple of size $d + 1$ where d is the size of the domain. Lewi et al. then extend this scheme to domains of size d^n. This results in right ciphertext tuples of size $dn + 1$. Our experimental results compare favourably with theirs, largely because the ciphertext sizes of Lewi et al.'s scheme are much larger.

The security of these ORE schemes is proven under a scenario similar to IND-OCPA [13] (see section 4.2.2). However, under realistic assumptions on what an adversary might do, these ORE schemes seem to have little security advantage over OPE schemes. For example, in $O(n \log n)$ comparisons an adversary can obtain a total ordering of the ciphertexts, and, hence the total ordering of the plaintexts. A disadvantage of ORE schemes are that they permit an equality test on ciphertexts [15, p.2] by using two comparisons. This could be used to aid a guessing attack on low-entropy plaintexts, e.g. [51]. A randomised OPE scheme, like ours, does not permit this. On the other hand, the information leakage of the ORE schemes so far proposed appears to be near-optimal.

To summarise, the OPE schemes presented in this paper differ from and improve on related work in four ways. First, they are, as far as we are aware, the only OPE schemes (as opposed to ORE) to be based on computationally hard problems (GACDP and DPolyACDP). Second, our schemes are randomised, rather than deterministic, so that ciphertexts of the same plaintext are different and randomly ordered. Third, our GACD based scheme is extremely efficient: only requiring $O(1)$ arithmetic operations to encrypt and decrypt. Likewise, our vector and polynomial based schemes are also efficient, although requiring more computation than the GACD based scheme. Finally, our schemes have near optimal bit leakage, as is the case for the ORE schemes discussed above. Popa et al.'s protocol [14], and similar work [37], have optimal bit leakage but require at least $O(n)$ communications.

[11, 28, 32, 51] describe some leakage-abuse attacks on OPE and ORE systems. We discuss such attacks in section 7.

4. An OPE scheme using Integer Approximate Common Divisors

Our OPE scheme is the symmetric encryption system (KGen, 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 $0 < m_1 \leq m_2 \leq \cdots \leq m_n \leq M$.

The scheme is conceptually simple and nondeterministic. To encrypt, we multiply a plaintext, m, by a large integer k, common to all ciphertexts i.e. the secret key, and then add a suitably large random integer r, unique to each ciphertext, to this product, where $r < k$ (see sections 4.0.1 and 4.1 for the bounds on r). The set of ciphertexts then forms an instance of the GACD problem (see section 4.1). To decrypt, we simply divide the ciphertext by k, ignoring any remainder. To see that the ciphertexts preserve the ordering of plaintexts, suppose we have two plaintexts m_1 and m_2, where $m_1 < m_2$. m_1 is encrypted as $c_1 = km_1 + r_1$, m_2 is encrypted as $c_2 = km_2 + r_2$. To preserve the ordering we require $c_2 - c_1 > 0$, i.e. $k(m_2 - m_1) > (r_1 - r_2)$. This follows, since the left hand side of the inequality is at least k, whereas the right hand side is at most $k-1$ (actually, it will be at most $\lfloor k - k^{3/4} \rfloor$, which is still less than k). We should note that if $m_1 = m_2$, i.e. we are encrypting a plaintext twice, then the order of the encryptions is random, since $\Pr(r_2 > r_1) \approx \frac{1}{2} - 1/k \approx \frac{1}{2}$, since $k \gg 1$.

4.0.1. Key Generation.

Both the security parameter space $\mathcal{S}$ and the secret key space $\mathcal{K}$ are the set of positive integers. Given a security parameter $\lambda \in \mathcal{S}$, with $\lambda > {}^8\!/_3 \lg M$, Algorithm 1 randomly chooses an integer $k \in [2^\lambda, 2^{\lambda+1})$ as the secret key, sk. So k is a $(\lambda+1)$-bit integer such that $k > M^{8/3}$ (see section 4.1). Note that k does not necessarily need to be prime.

Algorithm 1: Key Generation Algorithm KGen

Input : $\lambda \in \mathcal{S}, \lambda > {}^8\!/_3 \lg M$
Output : $k \in \mathcal{K}$
$k \leftarrow_\$ [2^\lambda, 2^{\lambda+1})$;
return k;

4.0.2. Encryption.

A plaintext $m_i \in \mathcal{M}$ is encrypted by Algorithm 2.

Algorithm 2: Encryption Algorithm Enc

Input : $m_i \in \mathcal{M}$
Input : $k \in \mathcal{K}$
Output : $c_i \in \mathcal{C}$
$r_i \leftarrow_\$ (k^{3/4}, k - k^{3/4})$;
$c_i \leftarrow m_i k + r_i$;
return c_i;

4.0.3. Decryption.

A ciphertext $c_i \in \mathcal{C}$ is decrypted by Algorithm 3

Algorithm 3: Decryption Algorithm Dec

Input : $c_i \in \mathcal{C}$
Input : $k \in \mathcal{K}$
Output : $m_i \in \mathcal{M}$
$m_i \leftarrow \lfloor c_i / k \rfloor$;
return m_i;

4.1. Security of the Scheme

Security of our scheme is given by the *general approximate common divisor problem* (GACDP), which is believed to be hard. It can be formulated [17, 23] as:

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

GACDP and *partial approximate common divisor problem* (PACDP), its close relative, are used as the basis of several cryptosystems, e.g. [24, 29, 64]. GACDP has been shown to be as hard as the "learning with errors" (LWE) problem [19], which is the basis of several post-quantum cryptosystems (examples include LIMA [61] and Lizard [21]). Solving the GACDP is clearly equivalent to breaking our system. 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 [48]. However, note that the model in [48] assumes that we are able to verify when a guess is correct, which does not seem to be the case here.

Howgrave-Graham [34] studied two attacks against GACDP, 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 [34], $\mathsf{GACD\_L}$, succeeds when $|x_0|, |y_0| < X = b_0^{\beta_0}$, and the divisor $d \geq b_0^{\alpha_0}$ and

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

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 $\sqrt{1 - \alpha_0 - \tfrac{1}{2}\alpha_0^2} \leq 1 - \tfrac{3}{2}\alpha_0$. This can only be satisfied if $\alpha_0 \leq 2/3$. But then squaring both sizes of the inequality implies $\alpha_0 \geq 8/11 > 2/3$, contradicting $\alpha_0 \leq 2/3$.

Suppose we take $\alpha_0 = 8/11$. Then, to foil this attack, we require $\beta_0 \geq 6/11$. For our system we have, $b_0 - a_0 = \max m_i - \min m_i = M$.[2] 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 $(k^{3/4},\, k - k^{3/4})$.

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

Cohn and Heninger [23] give an extension of Howgrave-Graham's algorithm to find the approximate divisor of m integers, where $m > 2$. Unfortunately, their algorithm is exponential in m in the worst case, though they say that it behaves better in practice. On the other hand, Chen and Nguyen [16, Appendix A] claim that Cohn and Heninger's algorithm is worse than brute force in some cases. In our case, the calculations in [23] do not seem to imply better bounds than those derived above.

We note also that the attack of Chen and Nguyen [17] 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 GACDP, see Galbraith et al. [31].

4.2. Security Models

It is obvious that any OPE cannot satisfy *indistinguishability under CPA* (IND-CPA) as a result of the ordering on ciphertexts. Furthermore, it can be argued that any notion of indistinguishability under CPA is not relevant to OPE in practice (see section 4.2.2). Various attempts have been made by Boldyreva and others [13, 14, 63, 67] to provide such indistinguishability notions. However, the security models impose practically unrealistic restrictions on an adversary. We discuss Boldyreva et al.'s notion of *indistinguishability under ordered CPA* (IND-OCPA) in section 4.2.2. However, while our usage scenario does not permit CPA and KPA, we do analyse our scheme according to Boldyreva et al.'s *window one-wayness* security model (see section 4.2.3), which we regard as reasonable. It should also be pointed out that satisfying an indistinguishability criterion does not guarantee that a cryptosystem is unbreakable, and neither does failure to satisfy it guarantee that the system is breakable.

4.2.1. One-Wayness.

A *one-way function* is a function which is easy to compute but it is hard to compute the inverse function on a random input. The one-wayness of the function $c(m) = km + r$ used by the scheme clearly follows from the assumed hardness of the GACD problem, since we avoid the known polynomial-time solvable cases.

4.2.2. IND-OCPA

The model in [13, p.6] and [44, p.20] is as follows:

Definition 2 (Indistinguishability under ordered CPA (IND-OCPA)). Given two equal-length sequences of plaintexts $(m_0^1 \dots m_0^q)$ and $(m_1^1 \dots m_1^q)$, where the m_b^j $(b \in [0,1], j \in [1,q])$ are distinct,[3] an adversary is allowed to present two plaintexts to a *left-or-right oracle* [8], $\mathcal{LR}^{(m_0,m_1,b)}$, which returns the encryption of m_b. The adversary is only allowed to make queries to the

[3] [13, p.6] and [44, p.20] do not clearly state this assumption but it appears that all plaintext values used must be distinct. This assumption clearly does not weaken the model.

oracle which satisfy $m_0^i < m_0^j$ iff $m_1^i < m_1^j$ for $1 \leq i, j \leq q$. The adversary wins if it can distinguish the left and right orderings with probability significantly better than $1/2$.

However, Boldyreva et al. [13, p.5] note, concerning chosen plaintext attacks: "in the symmetric-key setting a real-life adversary cannot simply encrypt messages itself, so such an attack is unlikely to be feasible". Further, they prove that no OPE scheme with a polynomial size message space can satisfy IND-OCPA. Lewi et al. [44] strengthen this result under certain assumptions.

The IND-OCPA model seems inherently rather impractical as a result of the requirement that an attacker makes queries to the oracle according to the condition $m_0^i < m_0^j$ iff $m_1^i < m_1^j$ for $1 \leq i, j \leq q$. A realistic analogue of an encryption oracle could not place such a restriction on an attacker. Additionally, an adversary with an encryption oracle could decrypt any ciphertext by bisection using $\lg M$ comparisons, where M is the size of the message space. Furthermore, Xiao and Yen [66] construct an OPE for the domain [1,2] and prove that it is IND-OCPA secure. However, this system is trivially breakable using a "sorting" attack [51].

For these reasons, we do not consider security models assuming CPA to be relevant to OPE.

4.2.3. Window One-Wayness.

We may further analyse our scheme under the same model as in [14], which was called *window one-wayness*. The scenario is as follows.

Definition 3 (Window one-wayness). An adversary is given the encryptions $c_1 \leq c_2 \leq \cdots \leq c_n$ of a sample of n plaintexts $m_1 \leq m_2 \leq \ldots \leq m_n$, chosen uniformly and independently at random from the plaintext space $[0, M)$. The adversary is also given the encryption c of a challenge plaintext m, and must return an estimate $\hat{m}$ of m and a bound r, such that $m \in (\hat{m} - r, \hat{m} + r)$ with probability greater than $1/2$, say. How small can r be so that the adversary can meet the challenge?

This model seems eminently reasonable, except for the assumption that the plaintexts are distributed uniformly. However, as we show in section 7.2, this assumption can be weakened in some cases for our scheme.

Since the m_i are chosen uniformly at random, a random ciphertext satisfies, for $\mathbf{c} \in [0, kM)$,

$$\Pr(\mathbf{c} = c) = \Pr(k\mathbf{m} + \mathbf{r} = km + r) = \Pr(\mathbf{m} = m) \Pr(\mathbf{r} = r) = \frac{1}{M}\frac{1}{k} = \frac{1}{Mk},$$

where $\mathbf{m} \leftarrow_{\$} [0, M)$, $\mathbf{r} \leftarrow_{\$} [0, k)$. Thus $\mathbf{c}$ is uniform on $[0, kM)$. Note that this is only approximately true, since we choose $\mathbf{r}$ uniformly from $[k^{3/4}, k - k^{3/4}]$. However, the total variation distance between these distributions is $2Mk^{3/4}/Mk = 2/k^{1/4}$. The difference between probabilities calculated using the two distributions is negligible, so we will assume the uniform distribution.

By assumption, the adversary cannot determine k by any polynomial time computation. So the adversary can only estimate k from the sample. Now, in a uniformly chosen sample $c_1 \leq c_2 \leq \cdots \leq c_n$ from $[0, kM)$, the sample maximum c_n is a sufficient statistic for the range kM, so all information about k is captured by c_n. So we may estimate k by $\hat{k} = c_n/M$. This is the maximum likelihood estimate, and is consistent but not unbiased. The minimum variance unbiased estimate is $(n+1)\hat{k}/n$, but using this does not improve the analysis, since the bias $k/(n+1)$ is of the same order as the estimation error, as we now prove. For any $0 \leq \varepsilon \leq 1$,

$$\begin{aligned}\Pr\left(\hat{k} \in k(1 \pm \varepsilon)\right) &\leq \Pr\left(c_n \geq kM(1-\varepsilon)\right) \\ &= 1 - (1-\varepsilon)^n \begin{cases} \leq n\varepsilon < 1/2 & \text{if } \varepsilon < 1/(2n); \\ \geq 1 - e^{-n\varepsilon} \geq 1/2 & \text{if } \varepsilon \geq \ln 2/n. \end{cases}\end{aligned}$$

Now, if $c = mk + r$, we can estimate m by $\hat{m} = c/\hat{k} \approx mk/\hat{k}$. Then

$$\Pr\left(m \in \hat{m}(1 \pm \varepsilon)\right) \approx \Pr\left(m \in mk/\hat{k}(1 \pm \varepsilon)\right) = \Pr\left(\hat{k} \in k(1 \pm \varepsilon)\right) < 1/2,$$

if $\varepsilon < 1/(2n)$. Thus, if $r \leq m/2n$, $\Pr(m \in \hat{m} \pm r) < 1/2$. Similarly, if $r \geq m \lg 2/n$, $\Pr(m \in \hat{m} \pm r) \geq 1/2$. Thus the adversary cannot succeed if $r \leq m/2n$, but can if $r \geq m \lg 2/n$.

It follows that only $\lg m - \lg(m/n) + O(1) = \lg n + O(1)$ bits of m are leaked by the system. However, $\lg n$ bits are leaked by inserting c into the sequence $c_1 \leq c_2 \leq \cdots \leq c_n$, so the leakage is close to minimal. By contrast the scheme of [13] leaks $1/2 \lg m + O(1)$ bits, independently of n. Therefore, by this criterion, the scheme given here is superior to that of [13] for all $n \ll \sqrt{M}$. Note that we have not assumed that m is chosen uniformly from $[0, M)$, but the leakage of the random sequence $c_1 \leq c_2 \leq \cdots \leq c_n$ is clearly $n \lg n - O(n)$ of the $M \lg M$ plaintext bits. This reveals little more than the $n \lg n$ bits already revealed by the known order $m_1 \leq m_2 \leq \cdots \leq m_n$.

4.3. A Vector-based Generalisation of the Scheme

We now detail an obvious generalisation of the above OPE scheme to n-vectors, for any ordering on vectors that respects the ordering on elements. We select a large integer k as the secret key using the key generation algorithm 1 as in the previous scheme. We encrypt a plaintext encoded as an n-vector $\vec{m}$ (see Algorithm 4) by multiplying all elements of $\vec{m}$ by the secret integer k and then adding a random integer in $(k^{3/4}, k - k^{3/4})$ to each element. Note that in this scheme, the elements of the vector ciphertexts form an instance of the GACD problem.

Algorithm 4: Encryption Algorithm Enc

Input : $\vec{m} \in \mathbb{Z}^n$
Input : $k \in \mathbb{Z}$
Output : $\vec{c} \in \mathbb{Z}^n$

$\vec{r} \leftarrow_{\$} (k^{3/4}, k - k^{3/4})^n$
$\vec{c} \leftarrow k\vec{m} + \vec{r}$
return $\vec{c}$

We decrypt by dividing each element of the ciphertext by k, ignoring the remainder, as shown in Algorithm 5 where $\lfloor\cdot\rfloor$ means round down each element of the vector.

Algorithm 5: Decryption Algorithm Dec

Input : $\vec{c} \in \mathbb{Z}^n$
Input : $k \in \mathbb{Z}$
Output : $\vec{m} \in \mathbb{Z}^n$
$\vec{m} \leftarrow \left\lfloor \frac{1}{k}\vec{c} \right\rfloor$
return $\vec{m}$

Security

As this system merely extends the earlier scheme to n-vectors, the analysis of 4.2 also applies to this vector-based scheme.

5. An OPE scheme using Polynomial Approximate Common Divisors

We now extend our approximate common divisor (ACD) approach to an OPE over polynomials which preserves the lexicographical ordering on polynomials. We can see this as a strengthening of the vector-based scheme in section 4.3, if the ordering on vectors is lexicographical. This alternate scheme is based on the related *decisional polynomial approximate common divisor* (DPolyACD) problem introduced in [20]. We redefine the problem here for clarity.

Definition 4 (Decisional polynomial ACD problem). Given a degree d polynomial $p(x)$, we have n approximate polynomial multiples of the form $p(x)q(x) + r(x)$ where $r(x)$ is a degree $d-1$ polynomial and $q(x)$ is a random polynomial from $\mathbb{Z}[x]$. Can we determine p(x) in a time polynomial in d, n, and the size of the coefficients of the polynomials involved?

We use this problem as the basis of our encryption scheme in the obvious way: we encrypt a plaintext encoded as a polynomial $m(x)$ as

$$k(x)m(x) + r(x)$$

where $k(x)$ is a degree d polynomial in $\mathbb{Z}[x]$ and $r(x)$ is a degree $d-1$ polynomial in $\mathbb{Z}[x]$. We decrypt by dividing the ciphertext polynomial $c(x)$ by $k(x)$ obtaining the quotient.

This construction preserves lexicographical ordering on polynomials. Let $m_1(x), m_2(x)$ be polynomials of degree at most n such that

$$m_1(x) = \sum_{j=0}^{n} m_{1j}x^j, \qquad m_2(x) = \sum_{j=0}^{n} m_{2j}x^j.$$

We have from our lexicographic ordering that $m_1(x) > m_2(x) \iff$ for some $l \leq n$,

$$\begin{aligned} m_{1j} &= m_{2j} \qquad (j > l) \\ m_{1l} &> m_{2l}. \end{aligned}$$

Let

$$\Delta(x) = m_1(x) - m_2(x) = \sum_{j=0}^{n} \delta_j x^j.$$

Then, because $m_{1l} > m_{2l}$, we have that the leading coefficient of $\Delta(x), \delta_l > 0$.

Lemma 1. *Let $k(x) = \sum_{i=0}^{d} k_j x^j$ have leading coefficient $k_d > 0$. Then*

$$k(x)m_1(x) > k(x)m_2(x) \iff m_1(x) > m_2(x).$$

PROOF. $m_1(x) > m_2(x) \iff \Delta(x) = m_1(x) - m_2(x)$ has leading coefficient $\delta_l > 0$. Thus $k(x)\Delta(x)$ has leading coefficient $k_d\delta_l > 0$. Therefore, $k(x)m_1(x) - k(x)m_2(x)$ has leading coefficient $k_d\delta_l > 0$, i.e. $k(x)m_1(x) > k(x)m_2(x)$. $\square$

It is easy to see that adding $r(x)$ to the product of $k(x)$ and $m(x)$ does not affect the ordering. Suppose we have degree $d-1$ polynomials $r_1(x)$ and $r_2(x)$. We can see that $r_1(x) - r_2(x)$ also has degree at most $d-1$. Therefore $k(x)m_1(x) + r_1(x) - (k(x)m_2(x) + r_2(x)) = k(x)\Delta(x) + (r_1(x) - r_2(x))$ still has leading coefficient $k_d\delta_l$ because x^{d+l} is a higher order term than x^{d-1} for $l \geq 0$. Note that if $l = 0$, i.e. $m_1(x)$ and $m_2(x)$ are identical, then the ciphertext ordering is random, as with the linear scheme presented in section 4.

This scheme could be used for very large arbitrary precision integer plaintexts. We can represent a large integer as the polynomial $\sum_{i=0}^{n} d_i b^i$, where b is the radix and the d_i are the digits. Also, it should be noted that the polynomial multiplication required by our scheme can be done using fast Fourier transform if the degrees of the plaintext and key polynomials are large.

However, there are problems with the basic scheme presented above which are addressed in section 5.1. We present a high level overview here.

Our error polynomial $r(x)$ has degree less than the degree of the product $k(x)m(x)$. This means that the leading term of the ciphertext is $k_d m_n x^{n+d}$. Given that $k_d m_n$ will not typically be a hard to factor integer, leaving this term in the ciphertext polynomial allows an attacker to factor its coefficient to obtain k_d and m_n. If there are several ciphertexts then it becomes easier to recover k_d by computing the gcd of the leading terms of each ciphertext. Therefore, we remove this highest order term from the ciphertext. In doing so, this leaves a problem of preserving the ordering on polynomials. Therefore, we set $k_d = 1$ and we re-introduce the leading coefficient of $m(x), m_n$, to the ciphertext by adding a (fixed) large multiple of m_n, km_n, to the coefficient of the next degree $n+d-1$ term. We choose k such that $k \gg m_i (i \in [0, n])$, where the m_i are the coefficients of the plaintext, and $k \gg k_j (j \in [0, d])$, where the k_j are the coefficients of the key polynomial. This large multiple ensures that the m_n terms dominate the ordering of the ciphertexts, since $km_n \gg m_i k_j, \forall i, j$. Furthermore, now the coefficients of the second highest order term of the ciphertext polynomials form an instance of the GACD problem ensuring that we do not forfeit security in preserving the ordering.

5.1. OPE Scheme

We encode an n-vector message $\vec{m} \in \mathbb{Z}_M^n$ as a polynomial $m(x)$ where its coefficients are the elements of $\vec{m}$. We choose a large integer k and a degree d polynomial $k(x) \in \mathbb{Z}[x]$ as the secret key as described in Algorithm 6.

Algorithm 6: Key Generation

Input: A security parameter, λ
Input: The degree of the key polynomial, d
Output: A secret key, $(k_1, k_2(x))$
$k_1 \leftarrow_{\$} [2^{\lambda-1}, 2^{\lambda})$
$k'(x) \leftarrow_{\$} \mathbb{Z}^{d-1}[x]$ `// Degree` $d-1$ `polynomial`
$k_2(x) \leftarrow x^d + k'(x)$
return $(k_1, k_2(x))$

We encrypt a plaintext using Algorithm 7. As described above, we remove the leading term of the polynomial product of $k(x)$ and $m(x)$ from the ciphertext. We also amend the coefficient of the next highest order term to preserve the ordering of plaintexts.

Algorithm 7: Encryption

Input: A plaintext encoded as the polynomial $m(x) = \sum_{i=0}^{n} m_i x^i$
Input: A secret key $(k_1, k_2(x))$
Output: A degree $n+d-1$ polynomial ciphertext $c(x)$
$p(x) = m_n x^{n+d} + \sum_{i=0}^{n+d-1} p_i x^i \leftarrow k_2(x) m(x)$
$p'(x) \leftarrow (p_{n+d-1} + k_1 m_n) x^{n+d-1} + \sum_{i=0}^{n+d-2} p_i x^i$
$r(x) \leftarrow_{\$} \mathbb{Z}^{d-1}[x]$ `// Degree` $d-1$ `polynomial`
$c(x) \leftarrow p'(x) + r(x)$
return $c(x)$

We decrypt using Algorithm 8. We denote returning only the quotient of polynomial division as $\lfloor p(x)/q(x) \rfloor$.

5.2. Security

The one-wayness of the system comes from the assumed hardness of the DPolyACD problem.

Algorithm 8: Decryption

Input: A ciphertext $c(x) = \sum_{i=0}^{n+d-1} c_i x^i$

Input: A secret key $(k_1, k_2(x))$

Output: A plaintext encoded as the polynomial $m(x) = \sum_{i=0}^{n} m_i x^i$

$m_n \leftarrow \lfloor c_{n+d-1}/k_1 \rfloor$

$c'(x) \leftarrow m_n x^{n+d} + (c_{n+d-1} - k_1 m_n)x^{n+d-1} + \sum_{i=0}^{n+d-2} c_i x^i$

$m(x) \leftarrow \lfloor c'(x)/k_2(x) \rfloor$

return $m(x)$

We also note that a ciphertext has $n+2d+1$ unknowns: the $n+1$ coefficients of $m(x)$, the d coefficients of $k(x)$ (since we set the leading coefficient to 1), and the d coefficients of $r(x)$. However, we only have $n+d$ linear equations, the coefficients of the ciphertext. Each new ciphertext adds a further $n+d+1$ unknowns: the $n+1$ coefficients of the plaintext polynomial and the d coefficients of the random polynomial, $r(x)$. Again this adds $n+d$ linear equations, meaning that we always have at least d more unknowns than equations. Furthermore, encrypting the same value twice does not yield any additional information to an attacker as the the second encryption only supplies d new equations, the lower order terms that have terms in $r(x)$ added to them, while adding a further d new unknowns, the coefficients of $r(x)$. Therefore, we still have more unknowns than equations. This makes the system of equations insoluble without knowledge of the coefficients of the secret polynomial $k_2(x)$. However, with the knowledge of the secret polynomial $k_2(x)$, we have and $n+d$ linear equations and $n+d$ unknowns ensuring it is possible to recover the plaintext from a ciphertext.

Like any OPE system, this system leaks information as a result of ordering. As with the linear scheme detailed earlier the bit leakage is $O(n \log n)$ by a similar proof.

6. Algorithms of Boldyreva type

We have chosen to compare our GACD-based scheme with that of Boldyreva et al. [13], since it has been used in practical contexts by the academic community [14, p.5], as well as in Popa et al.'s original version of CryptDB [55], which has been used or adopted by several commercial organisations [57]. However, scant computational experience with the scheme has been reported [55]. Therefore, we believe it is of academic interest to report our experimental results with respect to Boldyreva et al.'s scheme. We also discuss some simpler variants which have better computational performance. These are compared computationally with our scheme in section 8 below. The relative security of the schemes has been discussed above. Our vector and polynomial based schemes (sections 4.3 and 5) not not directly compare with [13], so we have not conducted our experiments using these variants.

In this section we describe generic encryption and decryption algorithms based on Boldyreva et al.'s algorithm [13], which sample from any distribution and which bisect on the domain (section 6.1). We also present an approximation of Boldyreva et al.'s algorithm which samples from the Beta distribution (section 6.2). The approximation and generic algorithms are used in our experimental evaluation presented in section 8.

6.1. Generic Algorithms

Algorithm 9 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 9 depends on a pseudorandom number generator, P, and a deterministic seed function, S. Likewise, Algorithm 10 constructs the inverse function $f^{-1} : \mathcal{C} \to \mathcal{M}$ so that $m = f^{-1}(c)$.

Algorithm 9: 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 `// 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`$(x, b, f(x), f(b), m)$
Initiate P with a fixed seed S_0
Choose $f(0), f(M)$ pseudorandomly so that $f(M) - f(0) > 3N/4$
return `RecursiveEncrypt`$(0, M, f(0), f(M), m)$

6.2. 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 10: 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`$(x, b, f(x), f(b), c)$
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 $c_i \le c \le c_{i+1}$.[4] Let $x = m - m_i$, $a = m_{i+1} - m_i - 1$, $y = c - c_i$, $b = c_{i+1} - c_i + 1$, so $1 \le x \le a$ and $0 \le y \le b$. 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 $0 \le y \le b$,

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

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 (1) is approximated by a continuous distribution with, for $0 \le z \le 1$,

$$\Pr\big(z \le \tilde{f}(x)/b < z + \mathrm{d}z\big) = \frac{z^{x-1}(1-z)^{a-x}}{\mathrm{B}(x, a-x+1)}\,\mathrm{d}z \tag{2}$$

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 $n \le M \le \sqrt{N}$,

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

is very small, so we will almost always need at least ${}^1\!/_3 \lg N$ bits of precision. Thus the approximation given by (2) remains good even when $a = 1$, since it is then the uniform distribution on $[0, b]$, where $b \ge N^{1/3}$ with high probability.

If $M = 2^r$, we will always have $a = 2^s$ and $x = 2^{s-1}$ in (2), so $a - x = x$, and (2) simplifies to

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

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

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

7. Leakage Abuse Inference Attacks

On information theoretic grounds, an OPE or ORE system inherently leaks around $\Omega(\log n)$ bits per ciphertext where n is the total number of ciphertexts. Naveed et al. [51] described a "sorting attack" on property preserving databases which exploits this leakage. In addition to sorting attacks which exploit the order-preserving property of OPE systems, an attacker is also able to exploit the frequency of plaintexts in a dataset to their advantage. Using auxiliary data, for example, the frequency of words in a piece of text, an attacker can use that distribution to recover plaintexts from a deterministic encryption system. As many OPE systems are deterministic, this makes frequency attacks especially effective. An extensive analysis of several OPE and ORE schemes is performed by Grubbs et al. in [32].

7.1. Sorting Attacks

We must assume $n \ll M$ to avoid the "sorting attack" of Naveed et al. [51]. Furthermore, to prevent the attack on low density datasets, we do not encrypt fields associated with common data, such as postal codes or first names, with OPE. This prevents the use of auxiliary data aiding cryptanalysis. Thus we are ruling out data where revealing the order reveals most of the information content. See section 7.3 for further discussion.

7.2. Frequency Attacks

To prevent a frequency attack, the source data requires sufficient entropy to provide negligible information to an attacker. Our system has the frequency hiding property of Kerschbaum [37] "built in" as a result of the random noise added to each ciphertext. However, this random noise is not sufficient to defeat frequency attacks for low entropy data. While the "noise" added to the ciphertext is in $[k^{3/4}, k - k^{3/4}]$ which is a large interval since $k \gg M$, for small entropy data, it is feasible to estimate the ranges of which ciphertexts correspond to a single plaintext value.

Consider the case that we knew a plaintext, ciphertext pair (m, c), even though our scenario does not permit it. Such a pair would not allow us to break the system, since $c/m = k + r/m \in [k, k + k/m]$, which is a large interval since $k \gg M$. We note that small values of m reveal much less information than large values. A number n of such pairs would give more information, but it still does not seem straightforward to estimate k closer than $\Omega\big(k/(M\sqrt{n})\big)$. Thus the system has some resistance to KPA, even though this form of attack is excluded by our model of single-party secure computation.

Flattening.

Where the source data does not have sufficient entropy to thwart a frequency attack, we can mitigate this by "flattening" the data. The flattening approach we use here is rather different from those in [2] and [69], though not completely unrelated.

Our scheme can be used in conjunction with any unknown increasing function $f(m)$ of m. If $f(m)$ is this function, then we encrypt m by $c = f(m)k + r$, where $r \leftarrow_{\$} (k^{3/4},\, k - k^{3/4})$, and decrypt by $m = f^{-1}\big(\lfloor c/k \rfloor\big)$. The disadvantage is that the ciphertext size will increase, but the entropy may be increased. Of course, if F is known, there will be little advantage. A particular, and useful, case of this is where the distribution function $F(m)$ of the plaintexts is known, or can be reasonably estimated. Then the distribution of the plaintexts can be "flattened" to an approximate uniform distribution on a larger set $[0, N)$, where $N \gg M$. Thus, suppose the distribution function $F(m)$ $(M \in [0, M))$ is known, and can be computed efficiently for given m. Further, we assume that $\Pr(\mathbf{m} = m) \geq 1/N$, so F is strictly increasing. This assumption is weak, since the probability that $\mathbf{m}$ is chosen to be an m with too small probability is at most M/N, which we assume to be negligible.

We interpolate the distribution function linearly on the real interval $\mathbb{R}[0, M)$, by $F(x) = (1 - u)F(m) + uF(m + 1)$ for $x = (1 - u)m + u(m + 1)$, where $u \in \mathbb{R}[0, 1)$. Then we will transform $m \in [0, M)$ randomly by taking $\tilde{m} = NF(x)$ where u is chosen randomly from the continuous uniform distribution on $\mathbb{R}[0, 1)$. It follows that $\tilde{m}$ is uniform on $\mathbb{R}[0, N)$, since F is increasing, and $\tilde{m} = NF(x)$, since

$$\Pr(\tilde{m} \leq y) = \Pr(x \leq F^{-1}(y/N)) = F(F^{-1}(y/N)) = y/N.$$

Now, since we require a discrete distribution, we take $\bar{m} = \lfloor \tilde{m} \rfloor$. We invert this by taking $\hat{m} = \lfloor F^{-1}(\bar{m}) \rfloor$. Now, since F is strictly increasing,

$$\hat{m} = \lfloor F^{-1}(\bar{m}/N) \rfloor \leq F^{-1}(\tilde{m}/N) < F^{-1}(NF(m + 1)/N) = m + 1$$

$$\hat{m} = \lfloor F^{-1}(\bar{m}/N) \rfloor > F^{-1}((\tilde{m} - 1)/N) \geq F^{-1}(NF(m - 1)/N) = m - 1,$$

and so $\hat{m} = m$. Thus the transformation is uniquely invertible. Of course, this does not imply that $\hat{m}$ and m will have exactly the same distribution, but we may also calculate

$$\Pr(\hat{m} \leq x) \leq \Pr(\bar{m} \leq NF(x)) < Pr(\tilde{m} \leq NF(x) + 1) = F(x) + 1/N,$$

$$\Pr(\hat{m} \leq x) \geq \Pr(\bar{m} < NF(x + 1)) \geq Pr(\bar{m} < NF(x)) = F(x).$$

This holds, in particular, for integers $x \in [0, M)$. Thus the total variation distance between the distributions of $\hat{m}$ and $\tilde{m}$ is at most M/N. Thus the difference between the distributions of $\tilde{m}$ and $\hat{m}$ will be negligible, since $N \gg M$.

This flattening allows us to satisfy the assumptions of the window one-wayness scenario above. The bit leakage in m is increased, however. The entropy has apparently been increased to $\lg N$, which allows us to handle relatively small plaintext spaces $[0, M)$, by expanding them to a larger space $[0, N)$. However, if an attacker has a good approximation to F, they can undo the transformation, and then this may be no better than our basic OPE scheme.

Let ρ denote the Shannon entropy of the data. Thus $\rho = -\sum_{m=1}^{M} p_m \lg p_m$, where $I(m) = -\lg p_m$ is the information content of m. We will assume that $F(m)$ has frequency function $p_m = F(m) - F(m-1)$.

Theorem 2. *Suppose we have a random sample of n from the plaintext distribution $\{p_m : m \in [M]\}$. If $\rho \gg \lg n$, the average entropy in the sorted random sample $m_1 < m_2 < \cdots < m_m$ is at least $\rho - \lg n$.*

Proof. The probability of the ordered sample is at most $n!$ times that of the unordered sample. Hence its entropy is at least $n\rho - \lg n! \geq n\rho - n\lg n$. Thus the average entropy is at least $\rho - \lg n$. □

Note that this is the same as the bit-leakage given in section 4.2.3. Note also that the same simple proof holds for other defintions of entropy, such as the min entropy, $-\max_{m=1}^{M} \lg p_m$.

Of course, if $\rho < \lg n$. The sample will contain repeated plaintexts, and the reduction in information content may be less then $\lg n$, but is hard to estimate, and will approach ρ as n increases. That is, there will be no entropy in the sorted data. To see that this actually happens, and could be a devastating vulnerability when F is known to an attacker, see section 7.3 below.

Finally, we will describe how this should be implemented. First we apply the OPE scheme once, with perhaps k not too large, to add some entropy to the data, by "frequency hiding" [37]. Determine F from this data, and apply the above transformation to make the data more uniform. Then apply the OPE scheme again, with a larger value of k to generate the ciphertext. We believe this would offer good security provided the data has sufficient entropy. Again, if F can be estimated closely by an attacker, this may not provide more protection than the basic scheme.

For very low-entropy data, there is little point in using OPE, since the order reveals almost everything. Consider, for example the 0/1 data used as an example in [37]. This has at most one bit of entropy. Using OPE, the only remaining unknown is what proportion of the data are 0's, and hence what proportion are 1's. If this ratio can be estimated, for example from similar plaintext data, then almost all the ciphertexts can be decrypted. The frequency hiding methods proposed in [37] provide no protection.

Theorem 3. *The increase in bit leakage for m as a result of this flattening is approximately $\lg(mp_m/F(m))$, where $p_m = F(m) - F(m-1)$.*

Proof. We will assume that $F(m)$ is a reasonably smooth distribution, so $F'(m)$ exists, and is approximately equal to the frequency function $p_m = F(m) - F(m-1)$. We have shown that $\tilde{m} = NF(m)$ is approximately uniform on $[0, N]$. Also, we have shown that we can estimate $\tilde{m}$ from $c(\tilde{m})$ only to within $\tilde{r} = \tilde{m}/n = NF(m)/n$. Thus we can estimate m to within r, where

$$NF(m)/n \approx NF(m+r) - NF(m) \approx rNF'(m) \approx rNp_m,$$

and hence $r \approx F(m)/np_m$. Thus the bit leakage is

$$\lg m - \lg(F(m)/np_m) = \lg n + \lg(mp_m/F(m)).$$

Thus the increase in bit leakage for m is approximately $\lg(mp_m/F(m))$. □

The leakage remains near-optimal for near-uniform distributions, where $\alpha/M \leq p_m \leq \beta/M$, for some constants $\alpha, \beta > 0$. In this case $\lg(mp_m/F(m)) \leq \lg(\beta/\alpha) = O(1)$. There are also distributions which are far from uniform, but the ratio $mp_m/F(m)$ remains bounded. Further, suppose we have a distribution satisfying $1/m^\alpha \leq p_m \leq 1/m^\beta$, for constants $\alpha, \beta > 0$ such that $0 < \alpha - \beta < 1/2$. Then $\lg(mp_m/F(m)) < 1/2 \lg m$, so the leakage is less than in the scheme of [13].

This transformation also allows us to handle relatively small plaintext spaces $[0, M)$, by expanding them to a larger space $[0, N)$.

Finally, note that the flattening approach here is rather different from those in [2] and [69], though not completely unrelated.

7.3. Published Attacks on OPE

Naveed et al. [51] describe four attacks on property-preserving encrypted data, particularly against the CryptDB system of Popa et al [55]. They are: *frequency analysis*, *l_p-optimisation*; *sorting attacks*; and *cumulative attacks*. Frequency analysis is a well known cryptanalytic attack against deterministic encryption systems. The frequencies of plaintexts are calculated from an auxiliary dataset and ordered. The frequencies of ciphertexts are calculated and ordered. The ciphertexts are then assigned to a plaintext such that the ith most frequent ciphertext is assigned to the ith most frequent plaintext. Where values have the same frequency, there is a potential for error in the assignment of ciphertext to plaintext. *l_p-optimisation* is a technique devised by Naveed et al. designed to minimise that error. After computing the frequencies of the ciphertexts, ψ, and auxiliary data, π, a permutation X of the plaintext frequencies is determined such that it minimises $\|\psi - X.\pi\|_p$, the l_p norm of the distance between the two. Once this permutation is computed, the permuted plaintext frequencies are used for frequency analysis. Note that our "flattening" technique discussed in section 7.2 is intended to combat this leakage.

Naveed et al. [51] denote an encrypted dataset as *δ-dense* where the space of unique plaintexts which relate to the n encryptions as dense forms a fraction δ of the overall message space. Given a *dense* ($\delta = 1$) dataset, and a deterministic OPE scheme, we can simply sort the ciphertexts and then assign each sorted ciphertext to its corresponding sorted plaintext value. Even with a randomised OPE scheme, by estimating the range of ciphertext that correspond to an encyption of a unique plaintext value we can still conduct this sorting attack. The auxiliary information is a large database that closely resembles the target database. Furthermore, Naveed et al. extend this attack to low density datasets ($0 < \delta < 1$) by the use of auxiliary information using their cumulative attack strategy which combines frequency analysis with a sorting attack. This attack additionally uses the *cumulative distribution functions* (CDFs) of the auxiliary data and ciphertexts using the observation that an OPE ciphertext value which is larger than some proportion of the other ciphertexts is likely to correspond to a plaintext value which is larger that the same proportion of the plaintext space. Therefore, the attack calculates the histogram and CDF of the ciphertexts and the histogram and CDF of the auxiliary data. As with *l_p-optimisation*, a permutation is computed that minimises the distance between the two histograms and the distance between the two CDFs. Using the permutation one can then match ciphertexts to plaintexts with high probability.

Grubbs et al. [32] improve on Naveed et al.'s cumulative attack by using an ordered matching technique. This seeks to avoid what they describe as "crossing", where a ciphertext c is matched to a plaintext p' and ciphertext c' is matched to plaintext p but $c < c'$ and $p < p'$ violating the ordering on the ciphertexts and plaintexts. They recast the cumulative attack as a graph problem where the graph $G = (U, V, E)$ is a bipartite graph with vertices U corresponding to ciphertexts, vertices V to plaintexts, and E the edges connecting the vertices of U to V. Each edge is labelled with a cost. A matching is a set of edges such that no common vertex is shared between edges in this set. Each matching is a decryption of some of the ciphertexts, therefore finding a minimal cost matching will be a good solution. The edge costs are the l_1 distance of frequencies. To be exact, for an edge (i, j), the cost $c(i, j)$ is:

$$c(i, j) = |\mathsf{H}_C(i) - \mathsf{H}_M(j)|,$$

where H_C and H_M are the histograms for the ciphertext and plaintext spaces respectively.

Durak et al. [28] extend Naveed et al.'s sorting and cumulative attacks for datasets, such as encrypted columns in a database, which are correlated. They conduct a sorting attack on each dataset individually and emit pairs (α_1, α_2) such that α_i is the matching for ciphertext c_i where $i = 1, 2$ corresponds to the dataset. They then construct functions which map ciphertexts to equally spaced points in the plaintext domain with in a set bound. These functions are then applied to the pairs generated from the sorting attack.

Onozawa et al. [54] offer a similar attack to Grubbs et al. but do not use the ordered matching optimisation technique given in Grubbs et al.

Bindschaedler et al. [11] use Bayesian inference with the *maximum likelihood estimator* (MLE) modelled by the multinomial distribution to compute ciphertext-plaintext mappings. They extend this Bayesian inference to attack correlated datasets.

The above attacks are characterised as "snapshot" attacks in the literature [43] because they rely on access to the entirety of the dataset. Attacks that abuse leakage from range query access patterns and result rankings such as those discussed by Kellaris et al. [36]and Lacharité et al. [43] are not particularly relevant to OPE, as the results from any range query on OPE encrypted data can be ordered. With an expected $O(n \log n)$ uniformly random queries, where n is the total number of unique items in the dataset, one can obtain the entirety of the dataset and then apply "snapshot" attacks, rendering these additional attacks redundant.

All of these attacks rely on auxiliary data. Grubbs et al. [32] use US census data and data from the 2016 Fraternal Order of Police breach [50] for their study. This data includes first names, last names, ages and zip codes. It is first encoded as a a big-endian base 27 integer where each character in a string is encoded as a base 27 digit. This encoded plaintext is then encrypted using several OPEs. The ciphertexts are then attacked using auxiliary data to assist cryptanalysis.

Table 1: Min entropies of source data used in Grubbs et al. [32]

Dataset	# 1st Names	Entropy	# Last Names	Entropy	Total Records
FOP (FOP)	3,862	4.52	116,677	6.74	621,662
California Muni (CALC)	3,777	4.76	59,935	7.02	255,956
Washington (WA)	3,525	4.78	67,206	7.30	228,934
Texas Compt. (TXCOM)	2,416	4.93	33,802	6.79	149,678
Florida (FL)	2,091	4.76	32,986	6.50	112,566
Maryland (MD)	2,551	4.68	36,698	6.61	111,183
Connecticut (CT)	2,016	4.38	30,623	7.42	77,613
New Jersey (NJ)	1,964	4.33	29,094	7.12	73,119
Iowa (IA)	1,734	4.67	22,616	7.14	60,035
Ohio (OH)	1,440	4.52	21,034	6.53	58,792
Texas A&M U. (TXAMU)	1,466	4.94	11,437	6.89	25,192
North Carolina (NCAR)	696	4.42	3,688	6.36	6,976
Illinois (IL)	243	4.52	1,021	6.98	1,259

For first names, they used a year by year tally of most popular male names from 1945 to 1993 gathered by the US Social Security Administration [62]. For last names, they used frequencies of last names of respondents for the 2000 US census. For ages, the US Census Bureau American Community Survey is used to construct a histogram of ages of respondents who specified their employment as "law enforcement". For zip codes, they use 2010 census data reporting population per zip code. The authors succeed in recovering almost all plaintexts from OPE encrypted first name and last name data.

However, the data that is used by Grubbs et al. is not of sufficient entropy to ensure that the OPE ciphertexts cannot be easily recovered. Table 1 shows the min entropies, measured in bits, for the datasets used in Grubbs et al.'s study. As can be seen from the table the first name data has a min entropy of 4 to 5 bits, equivalent to uniformly distributed data with 16 to 32 unique values. Similarly, the last name data has a min entropy of 6.5 to 7.3, equivalent to 91 to 158 unique uniformly distributed values. In a data set with 620,000 entries, we need entropy $\gg \lg 620,000 > 19$ for OPE to have reasonable protection. So a search key with 32 bits or more is required for security, not 4 bits. It is hardly surprising given the low entropy values that the plaintexts were easily recovered. This is particularly relevant to their attack on Popa et al.'s OPE protocol [56], where IND-CPA secure encryption can be used to encrypt the data. In that case, the encryption itself reveals very little information, so the success of the attack results entirely from the low entropy of the data.

The authors also discuss padding variable length data. However, the method they choose is to postpend data with spaces. While this method makes each plaintext, and, hence, ciphertext fixed length, it does not add any randomness to the data. To mitigate against frequency attacks, a better method, not discussed, would have been to pad with random data, suitably demarked, or to combine first and last names into a single string.

Similarly, Naveed et al.'s study [51] is conducted using data from the National Inpatient Sample (NIS) database of the US Healthcare Cost and Utilization Project (HCUP) [1]. However, as shown in Figure 1 of their paper, each field has a very small number of possible values (the greatest being day of the year from 1 to 365). As with the study of Grubbs et al., their data has low entropy which makes recovery of plaintexts significantly more probable.

Bindschaedler et al. [11] use the same datasets as Naveed et al. [51] and Grubbs et al. [32] and, hence, the same criticisms apply to their paper.

Durak et al. [28] use the latitude and longitude of California road intersections [45] along with location data gathered from a German politician, Malte Spitz, over 6 months [10]. They only use 2000 data points from the California dataset resulting in low entropy plaintexts. Similarly, the Spitz data is low entropy, with 1477 unique longitude-latitude pairs and 30,492 timestamps. Furthermore, their method has a large margin of error, up to 140km for the California dataset.

Additionally, we should point out that frequency analysis and ordering attacks are sophisticated "guessing" attacks. In all of the above studies, the accuracy of recovery is verified by comparing the estimate with the plaintext value. However, an attacker does not have access to the unencrypted data. If, as in the case for the attack on Kerschbaum's frequency hiding OPE [37], where a minority of the data is retrieved (30% for even the lowest entropy data set), how does the attacker know which records have been correctly inferred and which have not? This throws into question the efficacy of such attacks.

The conclusion that Grubbs et al. draw: that OPE should not be used whatsoever is too strong. It is clear from the studies discussed above that OPE should be used with care. In particular, it should not be used for low entropy data such as that used in those studies. However, Grubbs et al. concede that even inappropriate use of OPE is better than no encryption. We would maintain that, carefully applied, it is considerably better than no encryption.

8. Experimental Results

To provide a fair comparison with the majority of existing OPE schemes, such as [13], we have only conducted experimental evaluation of our GACD-based scheme. As mentioned earlier, our polynomial and vector-based schemes do not directly compare with schemes such as [13], etc. To evaluate the GACD-based scheme, we devised 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 [53]. Finally, the sorted ciphertexts were decrypted and it was verified that the plaintexts were also correctly sorted.

8.1. Small-Scale Cloud Implementation

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 [25] arbitrary precision integer classes. It is denoted as algorithm *GACD* in

Table 2: Timings for each experimental configuration ($n = 10000$). 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.

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
BCLO	7	143.72	191.48	70.78	154.01	192.42
BCLO	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

Figure 2: Encryption algorithm initialisation times

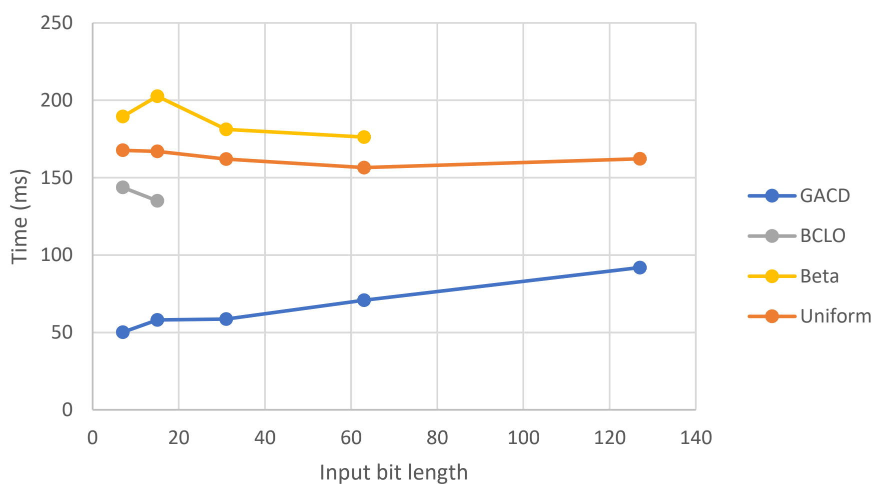


Table 2 and Figures 2 to 5. In addition, to provide comparison for our algorithm we have implemented Boldyreva et al.'s algorithm (referred to as *BCLO*) [13] along with two variants of the Boldyreva et al. algorithm. These latter variants are based on our generic version of Boldyreva et al.'s algorithm (see section 6.1).

Figure 3: Average encryption times

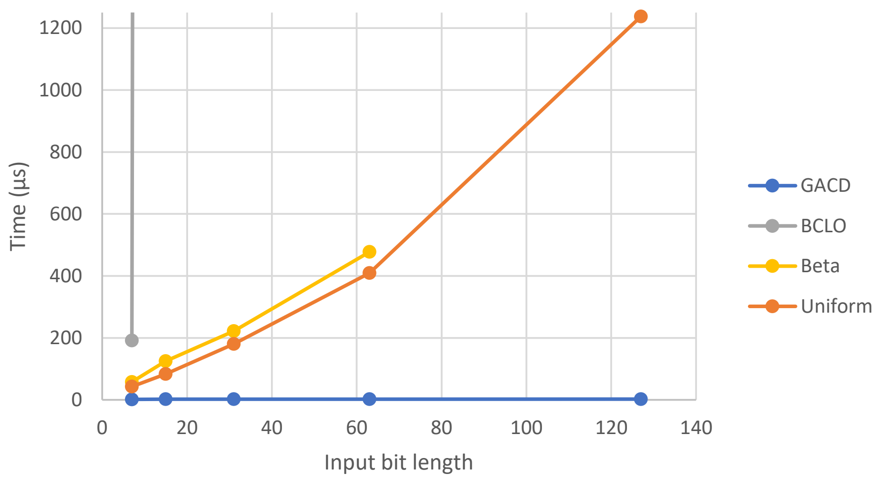
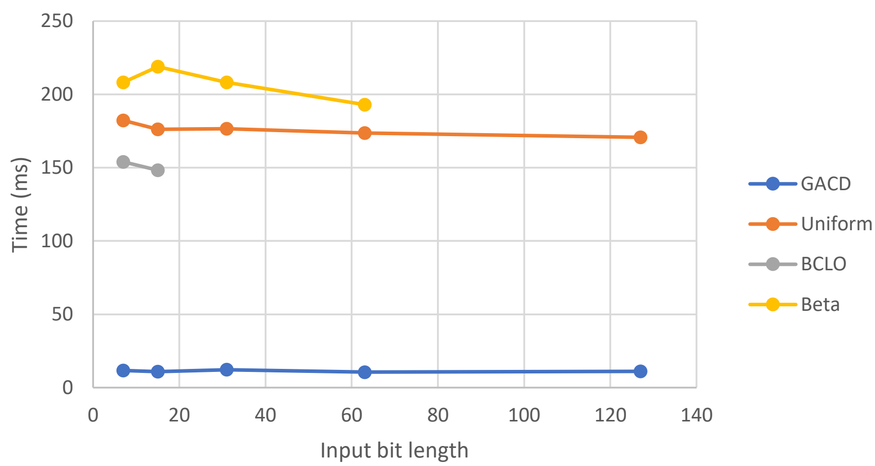
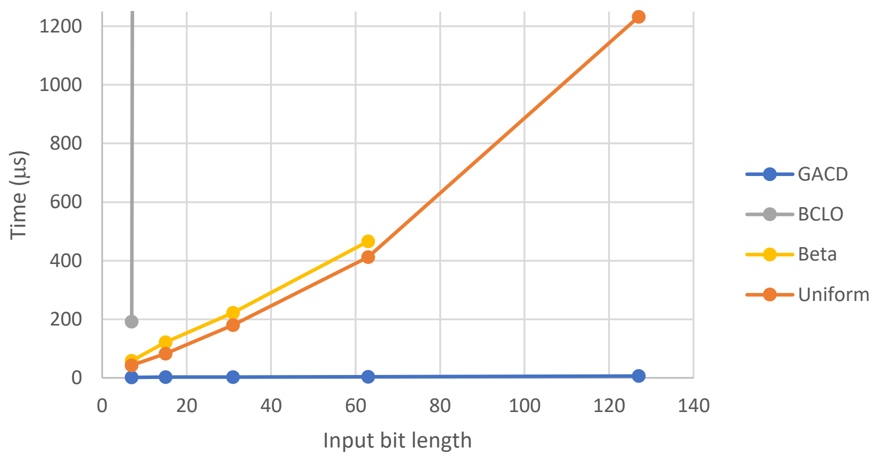


Figure 4: Decryption algorithm initialisation times



One is an approximation of Boldyreva et al.'s algorithm which samples ciphertext values from the Beta distribution (referred to as *Beta* in Table 2). The derivation of this approximation is given in section 6.2. The second samples ciphertexts from the uniform distribution (referred to as *Uniform* in Table 2). This variant appears in Popa et al.'s CryptDB [55] source code [57] as `ope-exp.cc`. The mean timings for each experimental configuration is tabulated in Table 2. 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

Figure 5: Average decryption times



generic Boldyreva algorithm. The Apache Commons Math [4] 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 2. To provide fair comparison, we have used similar configurations throughout. It should be pointed out that, for the *BCLO*, *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 4.1). So, our algorithm is more secure than *BCLO*, *Beta*, and *Uniform* for low entropy inputs.

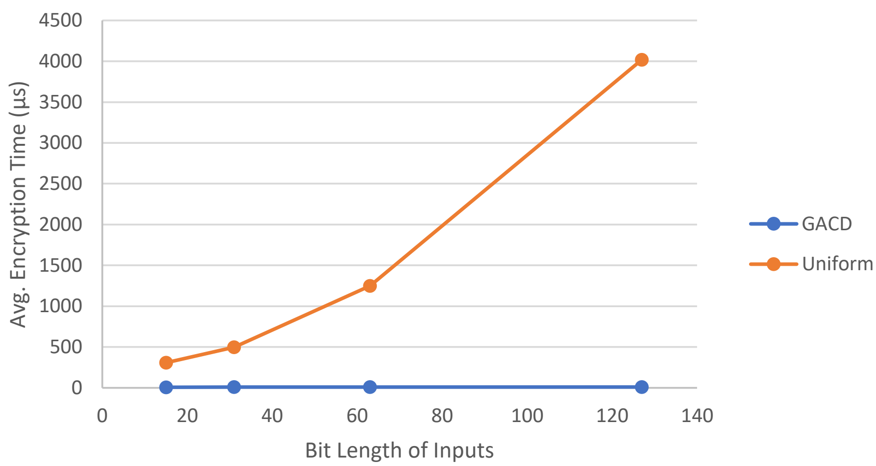
As shown by Table 2, our work compares very favourably with the other schemes. The encryption times of our algorithm outperform the next best algorithm (*Uniform*) by factors of 28 ($\rho = 7$) to 520 ($\rho = 127$). Furthermore, the decryption times grow sublinearly in the bit length of the inputs. 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 et al.'s version performs even worse. We believe this is down to the design of the algorithm, as stated in [13], which executes n recursions where n is the bit-size of the ciphertexts. 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. The algorithms based on Boldyreva et al. generate ciphertexts of double the bit size of the plaintexts, since we used a ciphertext space of size M^2 in our implementations,

Table 3: Timings for each experimental configuration ($n = 106272000$). ρ denotes the bit length of the unencrypted inputs. *Init* is the initialisation time for the encryption 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.

Algorithm	ρ	Encryption		MR Job	Dec.
		Init. (ms)	Enc. (μs)	Exec. (s)	(μs)
GACD	15	96.74	6.97	59.97	4.32
GACD	31	93.9	7.95	63.02	4.58
GACD	63	124.4	8.74	71.76	7.28
GACD	127	128.88	9.93	92.09	10.31
Uniform	15	71.24	307.71	56.53	280.22
Uniform	31	77.67	498.78	54.89	506.35
Uniform	63	66.74	1248.96	59.4	1324.25
Uniform	127	68.35	4018.86	69.02	4360.11

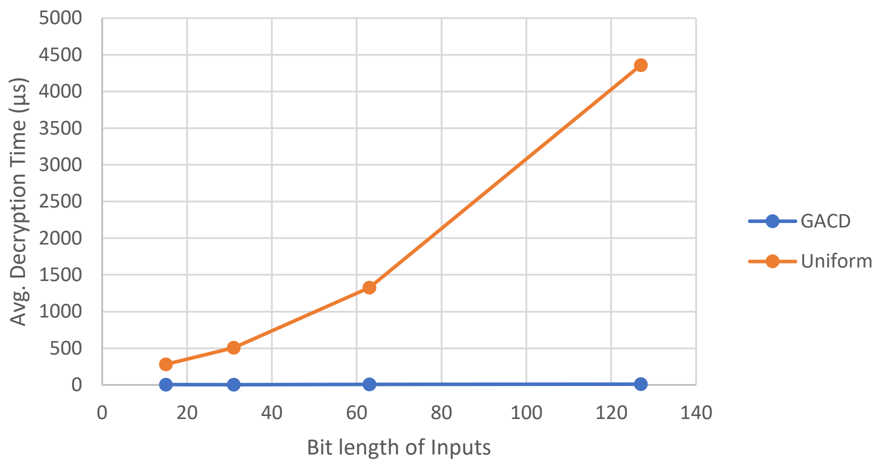
Figure 6: Average encryption times



where M is the size of the plaintext space. Our algorithm generates ciphertexts of length ~3.67 times the bit length of the ciphertext. However, Table 2 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 have shown above that the GACD approach gives a better theoretical guarantee of security than those of [13, 14, 63], which define security based on a game, rather than on the conjectured hardness of a known computational problem.

Figure 7: Average decryption times



8.2. Large-Scale Cloud Implementation: Microsoft Azure Cloud

We also scaled our experiment to a large HDInsight cluster in Microsoft's Azure cloud. Our experimental environment consisted of a HDInsight cluster comprising two D13v2 head nodes and 123 D4v2 worker nodes (984 worker cores). As a result of the large number of inputs (106,272,000), the input data was generated and encrypted using MapReduce programs running on the HDInsight cluster. In addition, we also had a MapReduce program to decrypt the data and verify that it had been correctly sorted.

For this experiment, we only performed the tests for our own OPE algorithm (GACD) and the variant of Boldyreva et al.'s algorithm sampling from the uniform distribution (Uniform). This was because it showed the best performance for encryption and decryption from our earlier experiment.

As one can see from Table 3 and Figures 6 and 7, the magnitude of the difference in performance between the two algorithms remains the same. In comparison with the results presented previously in this chapter, we note that, in both cases, the time to encrypt has increased approximately threefold and the time to decrypt twofold (GACD) and threefold (Uniform). This increase may be as a result of memory contention between map tasks running on the same worker node, particularly since the encryption process is memory intensive as a result of using arbitrary precision integers.

9. Conclusion

This paper has detailed several OPE schemes based on computationally hard problems. The schemes in section 4 are based on the general approximate common divisor problem (GACDP). The scheme presented in 5 is based on the decisional polynomial approximate common divisor problem (DPolyACDP).

These appear to be the first OPE schemes to be based on a computational hardness primitive, rather than a security game.

In section 8 we have reported on experiments to evaluate the practical efficacy of our first GACDP based scheme. We have compared this with the scheme of Boldyreva et al. [13]. Our vector and polynomial based schemes do not directly compare with the scheme of [13], so we do not extend the evaluation to these schemes. Our results show that the GACDP based scheme is very efficient, since is uses $O(1)$ arithmetic operations for encryption and decryption. As a trade-off against the time complexity of our algorithms, our scheme produces larger ciphertexts, ~ 3.67 times the number of bits of the plaintext. Furthermore, the results of 8.2 show that our scheme is scalable. However, as pointed out in section 8, ciphertext sizes had minimal impact on the running time of the MR job used in our experiments.

With regard to our stated purpose, our experimental results show that the efficiency of our scheme makes it suitable for practical computations in distributed computing environments such as the cloud.

In section 7 we have discussed several published attacks on OPE [11, 28, 32, 51, 54], particularly with regard to our own schemes. We note that our systems have the desirable frequency-hiding property "built in" as a result of the randomised encryption scheme. Furthermore, we show that these attacks are studied using encryptions of low entropy data. This low entropy data makes recovery of plaintexts highly probable yet no study has been performed on high entropy data. We conclude that the attacks of Naveed et al. [51], Grubbs et al. [32], and others do not prove that OPE should not be used but rather that it should be used with care. When carefully applied, it can prove to be a useful tool for sorting and searching on encrypted data.

Regarding future work, since our schemes seem promising, we intend to further investigate their practical deployment.

References

[1] Agency for Healthcare Research and Quality. Overview of the National (Nationwide) Inpatient Sample (NIS), 2009. URL `https://www.hcup-us.ahrq.gov/nisoverview.jsp`.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order Preserving Encryption for Numeric Data. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD 2004)*, pages 563–574. ACM, 2004. doi: 10.1145/1007568.1007632.

[3] C. Aguilar-Melchor, M.-O. Killijian, C. Lefebvre, T. Lepoint, and T. Ricosset. A Comparison of Open-Source Homomorphic Libraries With Multi-Precision Plaintext Moduli. WHEAT 2016, July 2016. URL `https://wheat2016.lip6.fr/ricosset.pdf`.

[4] Apache Software Foundation. Commons Math: The Apache Commons Mathematics Library, 2016. URL `http://commons.apache.org/proper/commons-math/`.

[5] A. Arasu, S. Blanas, K. Eguro, R. Kaushik, D. Kossmann, R. Ramamurthy, and R. Venkatesan. Orthogonal Security With Cipherbase. In *6th Biennial Conference on Innovative Data Systems Research (CIDR'13)*, 2013.

[6] L. Babai and L. Fortnow. Arithmetization: A new method in structural complexity theory. *computational complexity*, 1(1):41–66, 1991. doi: 10.1007/BF01200057.

[7] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A.-R. Sadeghi, G. Scerri, and B. Warinschi. Secure Multiparty Computation from SGX. In *Financial Cryptography and Data Security*, pages 477–497. Springer-Verlag, 2017. doi: 10.1007/978-3-319-70972-7_27.

[8] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997)*, pages 394–403. IEEE, 1997. doi: 10.1109/SFCS.1997.646128.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Proceedings of the 18th Annual Cryptology Conference (CRYPTO 1998)*, pages 26–45. Springer-Verlag, 1998. doi: 10.1007/BFb0055718.

[10] K. Biermann. Betrayed by our own data, 2011. URL `https://www.zeit.de/digital/datenschutz/2011-03/data-protection-malte-spitz`.

[11] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov. The Tao of Inference in Privacy-protected Databases. *Proc. VLDB Endow.*, 11(11):1715–1728, 2018. doi: 10.14778/3236187.3236217.

[12] T. Boelter, R. Poddar, and R. A. Popa. A Secure One-Roundtrip Index for Range Queries. Cryptology ePrint Archive, Report 2016/568, 2016.

[13] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-Preserving Symmetric Encryption. In *Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2009)*, pages 224–241. Springer-Verlag, 2009. doi: 10.1007/978-3-642-01001-9_13.

[14] A. Boldyreva, N. Chenette, and A. O'Neill. Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions. In *Proceedings of the 31st Annual Cryptology Conference (CRYPTO 2011)*, pages 578–595. Springer-Verlag, 2011. doi: 10.1007/978-3-642-22792-9_33.

[15] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman. Semantically Secure Order-Revealing Encryption: Multi-input Functional Encryption Without Obfuscation. In *Proceedings of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015), Part II*, pages 563–594. Springer-Verlag, 2015. doi: 10.1007/978-3-662-46803-6_19.

[16] 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, 2011.

[17] Y. Chen and P. Q. Nguyen. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers. In D. Pointcheval and T. Johansson, editors, *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2012)*, pages 502–519. Springer-Verlag, 2012. doi: 10.1007/978-3-642-29011-4_30.

[18] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. Practical Order-Revealing Encryption with Limited Leakage. In *Revised Selected Papers from the 23rd International Conference on Fast Software Encryption (FSE 2016)*, pages 474–493. Springer-Verlag, 2016. doi: 10.1007/978-3-662-52993-5_24.

[19] J. H. Cheon and D. Stehlé. Fully Homomophic Encryption over the Integers Revisited. In *Advances in Cryptology – EUROCRYPT 2015*, pages 513–536. Springer-Verlag, 2015.

[20] J. H. Cheon, H. Hong, M. S. Lee, and H. Ryu. The polynomial approximate common divisor problem and its application to the fully homomorphic encryption. *Information Sciences*, 326:41 – 58, 2016. doi: 10.1016/j.ins.2015.07.021.

[21] J. H. Cheon, D. Kim, J. Lee, and Y. Song. Lizard: Cut off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126, 2016.

[22] J. H. Cheon, K. Han, C. Lee, H. Ryu, and D. Stehlé. Cryptanalysis of the CLT13 Multilinear Map. *Journal of Cryptology*, 32(2):547–565, 2019. doi: 10.1007/s00145-018-9307-y.

[23] H. Cohn and N. Heninger. Approximate common divisors via lattices. In *Proceedings of the 10th Algorithmic Number Theory Symposium (ANTS-X)*, volume 1, pages 271–293. Mathematical Sciences Publishers, 2012. doi: 10.2140/obs.2013.1.271.

[24] J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In *Advances in Cryptology – CRYPTO 2011*, pages 487–504. Springer-Verlag, 2011. doi: 10.1007/978-3-642-22792-9_28.

[25] J.-M. Dautelle. JScience, Sept. 2014. URL `http://jscience.org`.

[26] K. Djemame, D. Armstrong, R. E. Kavanagh, J. Deprez, A. J. Ferrer, D. García-Pérez, R. M. Badia, R. Sirvent, J. Ejarque, and Y. Georgiou. TANGO: Transparent heterogeneous hardware Architecture deployment for eNergy Gain in Operation. *CoRR*, abs/1603.01407, 2016.

[27] L. Ducas and D. Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In E. Oswald and M. Fischlin, editors, *Proceedings of the 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015), Part I*, pages 617–640. Springer-Verlag, 2015. doi: 10.1007/978-3-662-46800-5_24.

[28] F. B. Durak, T. M. DuBuisson, and D. Cash. What else is revealed by order-revealing encryption? In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pages 1155–1166. ACM, 2016. doi: 10.1145/2976749.2978379. URL `http://doi.acm.org/10.1145/2976749.2978379`.

[29] J. Dyer, M. Dyer, and J. Xu. Practical Homomorphic Encryption Over the Integers for Secure Computation in the Cloud. In *Proceedings of the 16th IMA International Conference on Cryptography and Coding (IMACC 2017)*, volume 10655 of *Lecture Notes in Computer Science*, pages 44–76. Springer-Verlag, 2017. doi: 10.1007/978-3-319-71045-7_3.

[30] J. Dyer, M. Dyer, and J. Xu. Order-Preserving Encryption Using Approximate Integer Common Divisors. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 257–274. Springer-Verlag, 2017. doi: 10.1007/978-3-319-67816-0_15.

[31] S. D. Galbraith, S. W. Gebregiyorgis, and S. Murphy. Algorithms for the approximate common divisor problem. *LMS Journal of Computation and Mathematics*, 19(A):58–72, 2016. doi: 10.1112/S1461157016000218.

[32] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-Abuse Attacks against Order-Revealing Encryption. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 655–672, 2017. doi: 10.1109/SP.2017.44.

[33] M. Hoekstra, R. Lal, P. Pappachan, V. Phegade, and J. Del Cuvillo. Using Innovative Instructions to Create Trustworthy Software Solutions. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP 2013)*, pages 11:1–11:1. ACM, 2013. doi: 10.1145/2487726.2488370.

[34] N. Howgrave-Graham. Approximate Integer Common Divisors. In *Revised Papers from the International Conference on Cryptography and Lattices (CaLC 2001)*, pages 51–66. Springer-Verlag, 2001. doi: 10.1007/3-540-44670-2_6.

[35] H. Kadhem, T. Amagasa, and H. Kitagawa. MV-OPES: Multivalued-Order Preserving Encryption Scheme: A Novel Scheme for Encrypting Integer Value to Many Different Values. *IEICE Transactions on Information and Systems*, 93(9):2520–2533, 2010. doi: 10.1587/transinf.E93.D.2520.

[36] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*, pages 1329–1340. ACM, 2016. doi: 10.1145/2976749.2978386.

[37] F. Kerschbaum. Frequency-Hiding Order-Preserving Encryption. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 656–667. ACM, 2015. doi: 10.1145/2810103.2813629.

[38] F. Kerschbaum and A. Schroepfer. Optimal Average-Complexity Ideal-Security Order-Preserving Encryption. In *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS 2014)*, pages 275–286. ACM, 2014. doi: 10.1145/2660267.2660277.

[39] E. Khoury, M. Medlej, C. A. Jaoude, and C. Guyeux. Novel order preserving encryption scheme for wireless sensor networks. In *Proceeding of the 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM 2018)*, pages 1–6, April 2018. doi: 10.1109/MENACOMM.2018.8371028.

[40] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W.-H. Kim. Security of Stateful Order-Preserving Encryption. In *Information Security and Cryptology (ICISC 2017)*, pages 39–56. Springer-Verlag, 2018. doi: 10.1007/978-3-319-78556-1_3.

[41] S. F. Krendelev, M. Yakovlev, and M. Usoltseva. Order-preserving encryption schemes based on arithmetic coding and matrices. In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS 2014)*, pages 891–899. PTI, 2014. doi: 10.15439/2014F186.

[42] B. Kreuter, A. Shelat, and C.-H. Shen. Billion-Gate Secure Computation with Malicious Adversaries. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 285–300. USENIX, 2012.

[43] M. Lacharité, B. Minaud, and K. G. Paterson. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP 2018)*, pages 297–314, 2018. doi: 10.1109/SP.2018.00002.

[44] K. Lewi and D. J. Wu. Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, pages 1167–1178. ACM, 2016. doi: 10.1145/2976749.2978376.

[45] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *Advances in Spatial and Temporal Databases*, pages 273–290. Springer Berlin Heidelberg, 2005. doi: 10.1007/11535331_16.

[46] D. Liu and S. Wang. Programmable Order-Preserving Secure Index for Encrypted Database Query. In *Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD 2012)*, pages 502–509. IEEE, 2012. doi: 10.1109/CLOUD.2012.65.

[47] Z. Liu, X. Chen, J. Yang, C. Jia, and I. You. New order preserving encryption model for outsourced databases in cloud environments. *Journal of Network and Computer Applications*, 59:198 – 207, 2016. doi: 10.1016/j.jnca.2014.07.001.

[48] J. L. Massey. Guessing and Entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory (ISIT 1994)*, page 204. IEEE, 1994. doi: 10.1109/ISIT.1994.394764.

[49] V. Mavroudis, A. Cerulli, P. Svenda, D. Cvrcek, D. Klinec, and G. Danezis. A Touch of Evil: High-Assurance Cryptographic Hardware from Untrusted Components. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, pages 1583–1600. ACM, 2017. doi: 10.1145/3133956.3133961.

[50] K. McCarthy. US police contracts and private forum posts dumped online, 2016. URL `https://www.theregister.co.uk/2016/01/29/us_police_contracts_and_private_forum_posts_dumped_online/`.

[51] M. Naveed, S. Kamara, and C. V. Wright. Inference Attacks on Property-Preserving Encrypted Databases. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 644–655. ACM, 2015. doi: 10.1145/2810103.2813651.

[52] NIST Information Technology Laboratory. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, National Institute of Standards and Technology, 11 2001.

[53] O. O'Malley. TeraByte Sort on Apache Hadoop. Whitepaper, Yahoo, Inc., May 2008. URL `http://sortbenchmark.org/YahooHadoop.pdf`.

[54] S. Onozawa, N. Kunihiro, M. Yoshino, and K. Naganuma. Inference attacks on encrypted databases based on order preserving assignment problem. In *Advances in Information and Computer Security*, pages 35–47. Springer International Publishing, 2018. doi: 10.1007/978-3-319-97916-8_3.

[55] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011)*, pages 85–100. ACM, 2011. doi: 10.1145/2043556.2043566.

[56] R. A. Popa, F. H. Li, and N. Zeldovich. An Ideal-Security Protocol for Order-Preserving Encoding. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP 2013)*, pages 463–477. IEEE, 2013. doi: 10.1109/SP.2013.38.

[57] R. A. Popa, C. Redfield, S. Tu, H. Balakrishnan, F. Kaashoek, S. Madden, N. Zeldovich, and A. Burrow. CryptDB. Source code, 2014. URL `https://css.csail.mit.edu/cryptdb/`.

[58] H. Quan, B. Wang, Y. Zhang, and G. Wu. Efficient and Secure Top-k Queries With Top Order-Preserving Encryption. *IEEE Access*, 6:31525–31540, 2018. doi: 10.1109/ACCESS.2018.2847307.

[59] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.

[60] J. Saia and M. Zamani. Recent Results in Scalable Multi-Party Computation. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 24–44. Springer-Verlag, 2015. doi: 10.1007/978-3-662-46078-8_3.

[61] N. P. Smart, M. R. Albrecht, Y. Lindell, E. Orsini, V. Osheter, K. G. Paterson, and G. Peer. LIMA: A PQC Encryption Scheme. Technical report, University of Bristol, 2017. URL `https://lima-pq.github.io/files/lima-pq.pdf`.

[62] Social Security Administration. US social security name statistics, 2019. URL `https://www.ssa.gov/OACT/babynames/`.

[63] I. Teranishi, M. Yung, and T. Malkin. Order-Preserving Encryption Secure Beyond One-Wayness. In *Proceedings of the 20th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2014)*, pages 42–61. Springer-Verlag, 2014. doi: 10.1007/978-3-662-45608-8_3.

[64] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT 2010)*, pages 24–43. Springer-Verlag, 2010. doi: 10.1007/978-3-642-13190-5_2.

[65] M. Varia, S. Yakoubov, and Y. Yang. HETest: A Homomorphic Encryption Testing Framework. Cryptology ePrint Archive, Report 2015/416, 2015.

[66] L. Xiao and I.-L. Yen. A Note for the Ideal Order-Preserving Encryption Object and Generalized Order-Preserving Encryption. Cryptology ePrint Archive, Report 2012/350, 2012.

[67] L. Xiao and I.-L. Yen. Security analysis for order preserving encryption schemes. In *Proceedings of the 46th Annual Conference on Information Sciences and Systems (CISS 2012)*, pages 1–6. IEEE, 2012. doi: 10.1109/CISS.2012.6310814.

[68] C. Yang, W. Zhang, and N. Yu. Semi-order preserving encryption. *Information Sciences*, 387:266 – 279, 2017. doi: 10.1016/j.ins.2016.12.025.

[69] D. H. Yum, D. S. Kim, J. S. Kim, P. J. Lee, and S. J. Hong. Order-Preserving Encryption for Non-uniformly Distributed Plaintexts. In *Revised Selected Papers from the 12th International Workshop on Information Security Applications (WISA 2011)*, pages 84–97. Springer-Verlag, 2012. doi: 10.1007/978-3-642-27890-7_7.