---

**Proceedings Paper:**
Dyer, J, Dyer, M and Xu, J (2017) Practical Homomorphic Encryption Over the Integers for Secure Computation in the Cloud. In: Lecture Notes in Computer Science. 16th IMA International Conference on Cryptography and Coding (IMACC 2017), 12-14 Dec 2017, Oxford, UK. Springer, Cham , pp. 44-76. ISBN 9783319710440

https://doi.org/10.1007/978-3-319-71045-7_3

---

# Practical Homomorphic Encryption Over the Integers for Secure Computation in the Cloud

James Dyer[1], Martin Dyer[2], and Jie Xu[3]

[1] School of Computer Science, University of Manchester, UK.
james.dyer@postgrad.manchester.ac.uk
[2] School of Computing, University of Leeds, UK.
m.e.dyer@leeds.ac.uk
[3] School of Computing, University of Leeds, UK.
j.xu@leeds.ac.uk

**Abstract** We present novel homomorphic encryption schemes for integer arithmetic, intended primarily for use in secure single-party computation in the cloud. These schemes are capable of securely computing arbitrary degree polynomials homomorphically. In practice, ciphertext size and running times limit the polynomial degree, but this appears sufficient for most practical applications. We present four schemes, with increasing levels of security, but increasing computational overhead. Two of the schemes provide strong security for high-entropy data. The remaining two schemes provide strong security regardless of this assumption. These four algorithms form the first two levels of a hierarchy of schemes which require linearly decreasing entropy. We have evaluated these four algorithms by computing low-degree polynomials. The timings of these computations are extremely favourable by comparison with even the best of existing methods, and dramatically out-perform running times of directly comparable schemes by a factor of up to 1000, and considerably more than that for fully homomorphic schemes, used in the same context. The results clearly demonstrate the practical applicability of our schemes.

**Keywords:** Cryptography, symmetric encryption, homomorphic encryption, computing on encrypted data, secure computation in the cloud.

## 1 Introduction

With services like Amazon's Elastic MapReduce and Microsoft's HDInsight offering large-scale distributed cloud computing environments, computation in the cloud is becoming increasingly more available. Such services allow for computation on large volumes of data to be performed without the large investment in local computing resources. However, where the data that is processed is sensitive, such as financial or medical data, then uploading such data in its raw form to such a third-party service becomes problematic.

To take advantage of these cloud services, we require a means to process the data securely on such a platform. We designate such a computation, *secure computation in the cloud* (SCC). SCC should not expose input or output data to any other party, including the cloud service provider. Furthermore, the details

of the computation should not allow any other party to deduce its inputs and outputs. Cryptography seems the natural approach to this problem.

However, it should be noted that van Dijk and Juels [23] show that cryptography alone cannot realise secure *multi-party* computation in the cloud. Since our approach is via homomorphic encryption, we will restrict our attention to what we will call *secure single-party computation in the cloud* (SSCC).

*Homomorphic encryption* (HE) seems to offer a solution to the SSCC problem. First defined by Rivest et al. [50] in 1978, HE allows a function to be computed on encrypted inputs without ever decrypting the inputs. A *somewhat HE* scheme (SWHE) is a scheme which is homomorphic for only limited inputs and functions. *Fully HE* (FHE) is a scheme that is homomorphic for all functions and inputs. This was first realised by Gentry in 2009 [30], and appeared to be the ideal HE scheme.

However, despite the clear advantages of FHE, and many significant advances [12, 13], it remains largely impractical. Two implementations of FHE schemes, HELib [34] and FHEW [24], both perform very poorly in practice, both in their running time and space requirements (see section 2.6). Therefore, we take the view in this paper that only SWHE is, for the forseeable future, of practical interest. Our goal is to develop new SWHE schemes which are practically useful, and which we have tested with a realistic implementation.

In this paper, we present four novel SWHE schemes for encryption of integers that are additively and multiplicatively homomorphic. These schemes are capable of computing arbitrary degree polynomials. In section 2, we present our usage scenario, a summary of our results, and a discussion of related work. We present our initial homomorphic scheme in section 3, in two variants, HE1 and HE1N. HE1 (section 3.1) provides strong security for integers distributed with sufficient entropy. This security derives from the assumed hardness of the *partial approximate common divisor problem* (PACDP). HE1N (section 3.2) guarantees strong security for integers not distributed with sufficient entropy or where the distribution is not known, by adding an additional "noise" term. In addition to the hardness assumption, we prove that HE1N is IND-CPA secure [5]. Section 4 describes a further two variants, HE2 and HE2N, which increase the entropy of the plaintext by adding a dimension to the ciphertexts, which are 2-vectors. This further increases the security of these schemes by effectively doubling the entropy. HE2 (section 4.1) deals with integers of sufficient entropy, HE2N (section 4.2) with integers without the required entropy or of unknown distribution. HE2N also satisfies IND-CPA. We describe this in some detail, since it appears to be practically useful, and is the simplest version of our general scheme. We have performed extensive experimental evaluation of the four schemes presented in this paper. We report on this in section 5. Our results are extremely fa-

vourable when compared with other methods. In some cases, our algorithms outperform the running times of directly comparable schemes by a factor of up to 1000, and considerably more than that for fully homomorphic schemes, used in the same context. Finally, in section 6, we conclude the paper.

This paper also contains three appendices. In appendix A, we generalise HE2 and HE2N from 2-vectors to $k$-vectors, for arbitrary $k$, in the scheme HE$k$, with noisy variant HE$k$N. These schemes may also be practical for small enough $k$. In appendix B, we provide proofs of all theorems and lemmas in this paper. Finally, in appendix C, we provide the derivation of the bounds on the security parameters discussed in section 3.1.

## 2 Background

### 2.1 Scenario

As introduced above, our work concerns secure single-party computation in the cloud. In our scenario, a secure client wishes to compute a function on a large volume of data. This function could be searching or sorting the data, computing an arithmetic function of numeric data, or any other operation. We consider here the case where the client wishes to perform arithmetic computations on numeric data. This data might be the numeric fields within a record, with non-numeric fields being treated differently.

The client delegates the computation to the cloud. However, while the data is in the cloud, it could be subject to snooping, including by the cloud provider. The client does not wish to expose the input data, or the output of the computation, to possible snooping in the cloud. A snooper here will be a party who may observe the data and the computation in the cloud, but cannot, or does not, change the data or insert spurious data. (In our setting data modification would amount to pointless vandalism.) The snooping may be casual, displaying an uninvited interest, or malicious, intending to use data for the attacker's own purposes.

To obtain the required data privacy, the client's function will be computed homomorphically on an encryption of the data. The client encrypts the source data using a secret key and uploads the encryption to the cloud, with a homomorphic equivalent of the target computation. The cloud environment performs the homomorphic computation on the encrypted data. The result of the homomorphic computation is returned to the client, who decrypts it using the secret key, and obtains the output of the computation.

In this scenario, the source data is never exposed in the cloud, but encryptions of it are. A snooper may observe the computation of the equivalent homomorphic function in the cloud environment. As a result, they may be able to deduce what operations are performed, even though they do not know the inputs.

A snooper may also be able to inspect the (encrypted) working data generated by the cloud computation, and even perform side computations of their own. However, snoopers have no access to the secret key, so cannot make encryptions of their own.

### 2.2 Definitions and Notation

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

$\mathsf{KeyGen} : \mathcal{S} \to \mathcal{K}$ 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 $\mathcal{K}$, 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 $\mathcal{K}$, whose range is the plaintext space $\mathcal{M}$.

$\mathsf{Add} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ denotes the homomorphic addition function whose domain is $\mathcal{C}^2$ and whose range is $\mathcal{C}$.

$\mathsf{Mult} : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ denotes the homomorphic multiplication function whose domain is $\mathcal{C}^2$ and whose range is $\mathcal{C}$.

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

If $k^* = \binom{k+1}{2}$, $\boldsymbol{v}_\star = [v_1\ v_2\ \ldots\ v_{k^*}]^T$ denotes a $k^*$-vector which augments the $k$-vector $\boldsymbol{v} = [v_1\ v_2\ \ldots\ v_k]^T$ by appending elements $v_i = f_i(v_1, \ldots, v_k)$ $(i \in [k+1, k^*])$, for a linear function $f_i$. (All vectors are column vectors throughout.)

$\boldsymbol{e}_i$ denotes the $i$th unit vector $(i = 1, 2, \ldots)$, with size determined by the context.

$[x, y]$ denotes the integers between $x$ and $y$ inclusive, and $[x, y)$ denotes $[x, y] \setminus \{y\}$.

log denotes $\log_e$ and lg denotes $\log_2$.

If $\lambda$ is a security parameter, "with high probability" will mean with probability $1 - 2^{-\epsilon\lambda}$, for some constant $\epsilon > 0$.

Polynomial time or space will mean polynomial in the security parameter $\lambda$.

### 2.3 Formal Model of Scenario

We have $n$ integer inputs $m_1, m_2, \ldots, m_n$ distributed in $[0, M)$ according to a probability distribution $\mathcal{D}$. If $X$ is a random integer sampled from $\mathcal{D}$, let $\Pr[X = i] = \xi_i$, for $i \in [0, M)$. We will consider three measures of the *entropy* of $X$, measured in bits:

Shannon: $H_1(X) = -\sum_{i=0}^{M-1} \xi_i \lg \xi_i$, Collision: $H_2(X) = -\lg\left(\sum_{i=0}^{M-1} \xi_i^2\right)$,

Min: $H_\infty(X) = -\lg\left(\max_{i=0}^{M-1} \xi_i\right)$.
It is known that $H_1(X) \geq H_2(X) \geq H_\infty(X)$, with equality if and only if $X$ has the uniform distribution on $[0, M)$, in which case all three are $\lg M$. We will denote $H_\infty(X)$ by $\rho$, so it also follows that $H_1(X), H_2(X) \geq \rho$. We use the term "entropy" without qualification to mean min entropy, $H_\infty(X)$. Note that $H_\infty(X) = \rho \geq \lg M$ implies $\xi_i \leq 2^{-\rho}$, $i \in [0, M)$, and that $M \geq 2^\rho$.

We wish to compute a multivariate polynomial $P$ of degree $d$ on these inputs. A secure client $A$ selects an instance $\mathcal{E}_K$ of the encryption algorithm $\mathcal{E}$ using the secret parameter set $K$. $A$ encrypts the $n$ inputs by computing $c_i = \mathcal{E}_K(m_i)$, for $i \in [1, n]$. $A$ uploads $c_1, c_2, \ldots, c_n$ and $P'$ to the cloud computing environment, where $P'$ is the homomorphic equivalent of $P$ in the ciphertext space. The cloud environment computes $P'(c_1, c_2, \ldots, c_n)$. $A$ retrieves $P'(c_1, c_2, \ldots, c_n)$ from the cloud, and computes

$$P(m_1, m_2, \ldots, m_n) = {\mathcal{E}_K}^{-1}(P'(c_1, c_2, \ldots, c_n)).$$

A snooper is only able to inspect $c_1, c_2, \ldots, c_n$, the function $P'$, and the computation of $P'(c_1, c_2, \ldots, c_n)$, including subcomputations and working data, and perform side-computations on these.[4] Thus the snooper is *passive* or *honest-but-curious* [31].

### 2.4 Observations from Scenario

Our encryption schemes are essentially symmetric key encryption, though there is no key escrow or distribution problem. The public parameters of our schemes are exposed to the cloud, but they do not provide an encryption oracle.

Note that the $n$ inputs do not necessarily need to be uploaded at once, but $n$ is an upper bound on the total number of inputs. For example, if the polynomial is separable we might compute it in separate stages, and this might be useful in more dynamic situations.

This model is clearly susceptible to certain attacks. We consider ciphertext only, brute force, and cryptanalytic attacks. To avoid cryptanalytic attacks, we must choose the parameters of the system carefully. Here, a brute force attack will mean guessing the plaintext associated with a ciphertext. In our encryption schemes, it will be true that a guess can be verified. Since $\xi_i \leq 2^{-\rho}$ for $i \in [0, M)$, the expected number $\mu$ of guesses before making a correct guess satisfies $\mu \geq 2^\rho$. Massey [43] gave a corresponding result in terms of the Shannon entropy $H_1(X)$.

Similarly, probability of any correct guess in $2^{\rho/2}$ guesses is at most $2^{-\rho/2}$. This bound holds if we need only guess one of $n$ inputs, $m_1, m_2, \ldots, m_n$, even

[4] However, note that our "N" schemes below provide security even against more malicious snooping.

if these inputs are not independent. Therefore, if $\rho$ is large enough, a brute force attack is infeasible. An example of high entropy data is salaries for a large national or multinational business. Low entropy data might include enumerated types, such as gender.

In our model, known plaintext attack (KPA) is possible only by brute force, and not through being given a sample of plaintext, ciphertext pairs. Chosen plaintext attack (CPA) or chosen ciphertext attack (CCA) do not appear relevant to our model. Since $\mathcal{E}_K$ is never exposed in the cloud, there is no realistic analogue of an encryption or decryption oracle, as required by these attacks. In public key encryption, an encryption algorithm is available as part of the system, so CPA should be forestalled, though failure to satisfy IND-CPA [6] does not imply that we can break the system.

Following [5], it is common in studying symmetric key encryption to suppose that, in most practical settings, defence against CPA or CCA is necessary. While IND-CPA and IND-CCA are clearly desirable properties for a cryptosystem, their necessity, in the symmetric-key context, seems hard to justify. Both [4] and [9] provide examples intended to support this convention. However, these examples are unconvincing. Nevertheless, we show that the "N" variants of our HE schemes below do satisfy IND-CPA.

We note that observation of the function $P'$, which closely resembles $P$, might leak some information about its inputs. However, we assume that this information is far too weak to threaten the security of the system, as is common in the HE literature. However, if the threat is significant, "garbled circuits" [31] are a possible solution.

Finally, we note that our model of SSCC is very similar to the model of *private single-client computing*, described in [23]. Furthermore, they describe an example practical application, a privacy preserving tax return preparation program, which computes the relevant statistics on government servers without revealing the client's inputs. Another example, cited in [42], is a device which collects health data which is streamed to the cloud. Statistics are computed on the data and reported back to the device. To protect the patient's privacy this data is encrypted by the device and the computations are performed homomorphically. Erkin et al. [27] employ a similar scenario in the description of their privacy-preserving face recognition algorithm.

### 2.5 Our Results

We describe new practical HE schemes for the encryption of integers, to be employed in a SSCC system inspired by the HE scheme CryptDB [46]. CryptDB encrypts integers using the Paillier cryptosystem [45] which is additively homo-

morphic[5]. Similar systems ([52, 53]) use ElGamal [26] to support multiplications. The "unpadded" versions of these schemes must be used. These are not secure under CPA [32], reducing the advantage of a public-key system. These schemes do not support both addition and multiplication. Computing the inner product function requires re-encrypting the data once the multiplications have been done, so that the additions can be performed. In a SSCC system, this requires shipping the data back to the initiator for re-encryption, a significant communication overhead. We aim to support both addition and multiplication without this overhead. It should also be noted that a hybrid scheme of Paillier and ElGamal, for a given modulus, will be limited in the degree of polynomials that can be computed. Should a product or sum exceed the modulus then the result cannot be successfully decrypted.

Our scheme is inspired by the SWHE scheme of van Dijk et al. that is used as the basis for a public-key system. As in their system, we add multiples of integers to the plaintext to produce a ciphertext. However, [22] supports only arithmetic mod 2. We generalise their scheme to larger moduli.

We showed above that the input data must have sufficient entropy to negate brute force attacks. If the data lacks sufficient entropy, we will introduce more in two ways. The first adds random "noise" of sufficient entropy to the ciphertext, to "mask" the plaintext. This approach is employed in [22]. In our "N" variants below, we add a random multiple (from 0 to $\kappa$) of a large integer, $\kappa$, to the ciphertext, such that $m_i < \kappa$, for all $i \in [1, N]$. If the entropy of the original data was $\rho$, it becomes $\rho + \lg \kappa$. Therefore, if $\kappa$ is large enough, our data has sufficient entropy. But there is a downside. If the noise term grows too large, the ciphertext cannot be decrypted successfully. So we are restricted to computing polynomials of bounded degree, but this does not appear to be a practical problem.

The other technique will be to increase the dimension of the ciphertext. We represent the ciphertext as a $k$-vector, where each element is a linear function of the plaintext. Addition and multiplication of ciphertexts use linear algebra. The basic case $k = 1$ is described in section 3.1. Then we can increase the entropy by creating a $k$-vector ciphertext. Then we must guess $k$ plaintexts to break the system. Assuming that the inputs $m_1, m_2, \ldots, m_n$ are chosen independently from $\mathcal{D}$, and the entropy is $\rho$, the entropy of a $k$-tuple $(m_1, m_2, \ldots, m_k)$ is $k\rho$. Thus the $k$-vectors effectively have entropy $k\rho$. If $k$ is chosen large enough, we have sufficient entropy to prevent brute force attack. The assumption of independence among $m_1, m_2, \ldots, m_n$ can be relaxed, to allow some correlation, but we will not discuss the details. On the upside, some cryptanalytic attacks for $k = 1$ do not seem to generalise even to $k = 2$. The downside is that ciphertexts are

[5] Paillier supports computation of linear functions with known coefficients homomorphically by repeated addition

$k$ times larger, and each homomorphic multiplication requires $\Omega(k^3)$ time and space. For very large $k$, this probably renders the methods impractical. Therefore, we consider the case $k = 2$ in section 4. The general case is considered in appendix A.

Our work here supports computing arbitrary degree multivariate polynomials on integer data. However, we expect that for many practical applications, computing low-degree polynomials will suffice. See [42] for a discussion regarding this. In this paper, we present four variants of our scheme. Two provide strong security under the assumption that the input data has high entropy. The other two provide strong security regardless of this assumption. Appendix A generalises these four schemes to dimension $k$ ciphertexts.

### 2.6 Related Work

A comprehensive survey of somewhat and fully HE schemes is presented in [1]. In this section, we discuss those most related to our own work. Some related work ([46, 52, 53]) has already been discussed in section 2.5.

Our scheme is inspired by that of van Dijk et al. [22]. In their paper they produce an FHE scheme over the integers, where a simple SWHE scheme for modulo 2 arithmetic is "bootstrapped" to FHE. Our scheme HE1N below (section 3.2) may be regarded as a generalisation of theirs to arbitrary prime moduli. In van Dijk et al. [22], their symmetric scheme is transformed into a public key scheme. Though we could do this, we will not do so, since public key systems appear to have little application to our model. In [19], Coron et al. develop a similar encryption scheme, where the sum term in the ciphertext is quadratic rather than linear.

Several implementations of SWHE and FHE schemes have been produced. Lauter et al. [42] implement the SWHE scheme from [12]. However, they give results only for degree two polynomials. Our schemes are capable of computing degree three and four polynomials for practical key and ciphertext sizes. HELib [34] is an implementation of the BGV [13] FHE scheme. HELib-MP [48] is an adaptation of HELib to support multi-precision moduli. At the current time, it only supports basic SWHE features. The *Homomorphic Encryption Applications and Technology* (HEAT) project's *Homomorphic Encryption Application Programming Interface* (HE-API) [56] has currently integrated HELib and FV-NFLib [20], an implementation of the Fan and Vercauteren (FV) [28] SWHE scheme, under a single API. The authors appear to have made significant improvements in circuit evaluation times, but few details have been made available [10]. Microsoft's SEAL library [41] also implements the FV scheme, albeit, in a modified form. FHEW [25] implements the FHE scheme described in [24]. The performance of these implementations is discussed in section 5.

Erkin et al. [27] exploit the linearly-homomorphic properties of Paillier to compute feature vector matches in their privacy-preserving face recognition algorithm. Our schemes can likewise compute known linear functions, simply by not encrypting the coefficients of the function.

Catalano et al. [15] aim to extend a linearly-homomorphic system, such as Paillier [45], to compute multivariate quadratics homomorphically. However, their extension relies on pre-computing a product for each pair of plaintexts and then applying a linear function on the encryption of these products. As such, it does not extend the underlying linear encryption scheme and is not multiplicatively homomorphic. They claim that their system can compute any degree 2 polynomial with at most one multiplication. However, it is not clear how they would compute the polynomial $m_1 \cdot (m_2 + \ldots + m_n)$ without performing $n-1$ offline multiplications. By contrast, our scheme would only require one multiplication. In [14], Catalano et al. extend their approach to cubics.

Zhou and Wornell [59] construct a scheme based on integer vectors, similar, in some respects, to our HE2 (section 4.1) and HE$k$ (appendix A) schemes. Bogos et al. [8] demonstrate that the system displays some theoretical insecurities. However, the question of whether these are of practical importance is not addressed.

The symmetric MORE scheme [39] uses linear transformations, as do our schemes but in a different way. MORE has been shown [57] to be insecure against KPA, at least as originally proposed. However, whether KPA is relevant in applications of the scheme is unclear.

Recent work on *functional encryption* [33] should also be noted. While these results are of great theoretical interest, the scenario where such schemes might be applied is rather different from our model. Also, the methods of [33] seem too computationally expensive to be of practical interest in the immediate future.

## 3 Initial Homomorphic Scheme

### 3.1 Sufficient Entropy (HE1)

We have integer inputs $m_1, m_2, \ldots, m_n \in [0, M)$. (Negative integers can be handled as in van Dijk et al. [22], by taking residues in $[-(p-1)/2, (p-1)/2)$, rather than $[0, p)$.) We wish to compute a polynomial $P$ of degree $d$ in these inputs. The inputs are distributed with entropy $\rho$, where $\rho$ is large enough, as discussed in section 2.3 above. In practical terms, $\rho \geq 32$ will provide sufficient entropy for strong security, since breaking the system would require more than a billion guessses. Our HE scheme is the system $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Add}, \mathsf{Mult})$.

**Key Generation.** Let $\lambda$ be a security parameter, measured in bits. Let $p$ and $q$ be randomly chosen large distinct primes such that $p \in [2^{\lambda-1}, 2^{\lambda}]$, and $q \in [2^{\eta-1}, 2^{\eta}]$, where $\eta \approx \lambda^2/\rho - \lambda$. Here $\lambda$ must be large enough to negate

direct factorisation of $pq$ (see [40]), and $p$ and $q$ are chosen to negate Coppersmith's attack [18]. We will also require $p > (n+1)^d M^d$ to ensure that $P(m_1, m_2, \ldots, m_n) < p$, so that the result of the computation can be successfully decrypted. Our bounds are worst case, allowing for polynomials which contain all possible monomial terms. For some applications, they will be much larger than required to ensure that $P(m_1, m_2, \ldots, m_n) < p$ and smaller bounds will suffice. Our function KeyGen will randomly select $p$ and $q$ according to these bounds. Then $p$ is the private symmetric key for the system and $pq$ is the modulus for arithmetic performed by Add and Mult. $pq$ is a public parameter of the system. We assume that the entropy $\rho \gg \lg \lambda$, so that a brute force attack cannot be carried out in polynomial time.

**Security Parameters.** We can easily set the security parameters $\lambda$ and $\eta$ to practical values. If $n \approx \sqrt{M}$, $M \approx 2^\rho$ then we may take $\lambda \approx 3d\rho/2$ and $\eta \approx 3d\lambda/2 - \lambda$ (see appendix C). For, example, if $\rho = 32$, $d = 4$, we can take any $\lambda > 192$, $\eta > 960$.

**Encryption.** We encrypt a plaintext integer $m$ as

$$\mathsf{Enc}(m, p) = m + rp \pmod{pq}, \text{ where } r \xleftarrow{\$} [1, q).$$

**Decryption.** We decrypt the ciphertext $c$ by $\mathsf{Dec}(c, p) = c \pmod{p}$.

**Addition.** The sum modulo $pq$ of two ciphertexts, $c = m + rp$ and $c' = m' + r'p$, is

$$\mathsf{Add}(c, c') = c + c' = m + m' + (r + r')p \pmod{pq}.$$

This decrypts to $m + m'$, provided $m + m' < p$.

**Multiplication** The product modulo $pq$ of two ciphertexts, $c = m + rp$ and $c' = m' + r'p$, is

$$\mathsf{Mult}(c, c') = cc' = mm' + (rm' + r'm + rr'p)p \pmod{pq},$$

which decrypts to $mm'$, provided $mm' < p$.

**Security.** Security of the system is provided by the *partial approximate common divisor problem* (PACDP), first posed by Howgrave-Graham [36], but can be formulated [16, 17] as:

**Definition 1.** *(Partial approximate common divisor problem.) Suppose we are given one input $x_0$, of the form $pr_0$, and $n$ inputs $x_i$, of the form $pr_i + m_i$, $i \in [1, n]$, where $p$ is an unknown constant integer and the $m_i$ and $r_i$ are unknown integers. We have a bound $B$ such that $|m_i| < B$ for all $i$. Under what conditions on the $m_i$ and $r_i$, and the bound $B$, can an algorithm be found that can uniquely determine $p$ in time polynomial in the total bit length of the numbers involved?*

A straightforward attack on this problem is by brute force. Consider $x_1$. Assuming that $m_1$ is sampled from $\mathcal{D}$, having entropy $\rho$, we successively try values for $m_1$ and compute $\gcd(x_0, x_1 - m_1)$ in polynomial time until we find a divisor that is large enough to recover $p$. Then we can recover $m_i$ as $(x_i \bmod p)$ for $i \in [2, n]$. As discussed in section 2.3, the search will requires $2^\rho$ gcd operations in expectation. Note that publicly known constants, need not, and should not be encrypted. Encrypting them provides an obvious guessing attack.

Several attempts have been made to solve the PACDP [16, 17, 36], resulting in theoretically faster algorithms for some cases of the problem. The paper [16] gives an algorithm requiring only $\sqrt{M}$ polynomial time operations if $\mathcal{D}$ is the uniform distribution on $[0, M)$, and hence $\rho = \lg M$. No algorithm running in time subexponential in $\rho$ is known for this problem, so the encryption will be secure if $\rho$ is large enough. See [29] for a survey and evaluation of attacks on PACDP.

Our system is a special case of PACDP, since we use the residues modulo a distinct semiprime. A semiprime is a natural number that is the product of two primes. A distinct semiprime is a semiprime where the primes are distinct. We call this the *semiprime partial approximate common divisor problem* (SPACDP). It is a restriction, but there is no reason to believe that it is any easier than PACDP.

**Definition 2.** *(Semiprime factorisation problem.) Given a semiprime $s$, the product of primes $p$ and $q$, can $p$ and $q$ be determined in polynomial time?*

The computational complexity of this problem, which lies at the heart of the widely-used RSA cryptosystem, is open, other than for quantum computing, which currently remains impractical. We will show that breaking HE1 is equivalent to semiprime factorisation. Therefore, our scheme is at least as secure as unpadded RSA [49].

**Theorem 1.** *An attack against HE1 is successful in polynomial time if and only if we can factorise a distinct semi-prime in polynomial time.*

With low entropy plaintexts, there is a brute force attack on this system, which we call a *collision attack*. Suppose we have a pair of equal plaintexts $m_1 = m_2$. The difference between their encryptions $(c_1 - c_2)$ is an encryption of 0, and KPA is possible. In fact, for $n$ plaintexts $m_1, m_2, \ldots, m_n$, if there exist $i, j \in [1, n]$ with $m_i = m_j$, then $\prod_{1 \le i < j \le n}(c_j - c_i)$ is an encryption of 0. However, if there is sufficient entropy, this attack is not possible.

**Lemma 1.** *If the inputs $m$ have entropy $\rho$ then, for any two independent inputs $m_1, m_2$, $\Pr(m_1 = m_2) \le 2^{-\rho}$.*

Thus, for $n$ inputs, $m_1, m_2, \ldots, m_n$ the probability that there exist $i, j \in [1, n]$ with $m_i = m_j$ is at most $\binom{n}{2} 2^{-\rho}$. If $n < 2^{-\rho/3}$, this probability is at most $2^{-\rho/3}$. Hence, for large enough $\lambda$, collision attack is infeasible.

### 3.2 Insufficient Entropy (HE1N)

Suppose now that the integer inputs $m_i, i \in [1, n]$, are distributed with entropy $\rho$, where $\rho$ is not large enough to negate a brute force guessing attack. Therefore, we increase the entropy of the plaintext by adding an additional "noise" term to the ciphertext. This will be a multiple $s$ (from 0 to $\kappa$) of an integer $\kappa$, chosen so that the entropy $\rho' = \rho + \lg \kappa$ is large enough to negate a brute force guessing attack. As a result of the extra linear term in the ciphertext, we compute $P(m_1, \ldots, m_n, \kappa)$ instead. We can easily retrieve $P(m_1, \ldots, m_n)$ from $P(m_1, \ldots, m_n, \kappa)$.

**Key Generation.** KeyGen now randomly chooses $p$ and $q$ as in HE1, but with $\eta = \lambda^2/\rho' - \lambda$, and $p > (n+1)^d (M + \kappa^2)^d$ so that $P(m_1 + s_1\kappa, m_2 + s_2\kappa, \ldots, m_N + s_n\kappa) < p$, when $s_1, s_2, \ldots, s_n \in [0, \kappa)$. KeyGen also randomly chooses $\kappa$, where $\kappa > (n+1)^d M^d$, so that $P(m_1, m_2, \ldots, m_n) < \kappa$. The secret key, sk, is now $(\kappa, p)$.

**Security Parameters.** Again, we can set the security parameters $\lambda$ and $\eta$ to practical values. If we assume $M \approx 2^\rho$ and large enough $n$, as in section 3.1, then we may take $\lg \kappa > d(\lg n + \rho)$, $\rho' = \rho + \lg \kappa$, $\lambda > d(\lg n + 2 \lg \kappa)$. Then, for example, if $d = 3$, $\lg n = 16$, $\rho = 8$, then $\lg \kappa > 72$, $\rho' = 80$, $\lambda > 480$, $\eta > 2400$. In the extreme case that the inputs are bits, so $\rho = 1$, and $d = 3$, $\lg n = 16$, then we can take $\lg \kappa \approx 51$ and $\rho' \approx 52$, and we have $\lambda > 354$, $\eta > 2056$, which is only 15% smaller than for $\rho = 8$.

**Encryption.** We encrypt plaintext $m$ as $\mathsf{Enc}(m, \mathsf{sk}) = m + s\kappa + rp \pmod{pq}$, where $r \xleftarrow{\$} [1, q)$ and $s \xleftarrow{\$} [0, \kappa)$.

**Decryption.** We decrypt ciphertext $c$ as $\mathsf{Dec}(c, \mathsf{sk}) = (c \bmod p) \bmod \kappa$.

**Arithmetic.** Addition and multiplication of ciphertexts is as above.

**Security.** The use of random noise gives the encryption the following "indistinguishability" property, which implies that the system satisfies IND-CPA [5, 6].

**Theorem 2.** *For any encryption $c$, $c \bmod \kappa$ is polynomial time indistinguishable from the uniform distribution on $[0, \kappa)$. Thus HE1N satisfies IND-CPA, under the assumption that SPACDP is not polynomial time solvable.*

Therefore, HE1N is resistant to both the "guessing" and "collision" attacks discussed in section 3.1.

**Hybrid scheme.** Note that mixed data, some of which has high entropy and some low, can be encrypted with a hybrid of HE1 and HE1N. More generally, we can choose $s$ to be smaller for higher entropy and larger for lower entropy, say $s \in [0, \chi_i)$, where $0 \leq \chi_i < \kappa$, for the $i$th data type, rather than $[0, \kappa)$. However, $\kappa$ itself remains the same for all $i$, or we cannot decrypt. Then the entropy increases to $\rho_i + \lg \chi_i$ for data type $i$. The advantage is a smaller blow-up in the noise. A possible disadvantage is that this mixed scheme may not necessarily have the IND-CPA property of Theorem 2. The same idea can be applied to HE2 and HE2N below, and to the HE$k$N schemes, for $k > 2$, described in the appendix.

## 4 Adding a Dimension

In this section we discuss adding an additional dimension to the ciphertext, which becomes a 2-vector. The purpose of this is to increase the level of security beyond HE1 and HE1N. In both schemes presented below, HE2 and HE2N, we add a further vector term, with two further secret parameters. The two schemes presented below have a constant factor overhead for arithmetic operations. An addition operation in the plaintext space requires two additions in the ciphertext space, and a multiplication in the plaintext space requires nine multiplications and four additions in the ciphertext space.

### 4.1 Sufficient Entropy (HE2)

As with HE1, it is assumed that the inputs $m_i$ $(i \in [1, n])$ are of sufficient entropy.

**Key Generation.** $p$ and $q$ are randomly chosen by KeyGen according to the bounds given in section 3.1. KeyGen sets $\boldsymbol{a} = [a_1\ a_2]^T$, where $a_i \xleftarrow{\$} [1, pq)$ $(i \in [1, 2])$ such that $a_1, a_2, a_1 - a_2 \neq 0 \pmod p$ and $\bmod q$).[6] KeyGen also sets $R$, the re-encryption matrix (see "Multiplication") as

$$\begin{bmatrix} 1 - 2\alpha_1 & \alpha_1 & \alpha_1 \\ -2\alpha_2 & \alpha_2 + 1 & \alpha_2 \end{bmatrix},$$

where

$$\alpha_1 = \beta^{-1}(\sigma a_1 + \varrho p - a_1^2), \qquad \alpha_2 = \beta^{-1}(\sigma a_2 + \varrho p - a_2^2), \tag{1}$$

such that $\beta = 2(a_2 - a_1)^2$, $\varrho \xleftarrow{\$} [0, q]$ and $\sigma \xleftarrow{\$} [0, pq)$.

The secret key sk is $(p, \boldsymbol{a})$ and the public parameters are $pq$ and $R$.

[6] The condition $a_1, a_2, a_1 - a_2 \neq 0$, $(\bmod p, \bmod q)$ fails with exponentially small probability $3(1/p + 1/q)$. Thus, $a_1$ and $a_2$ are indistinguishable in polynomial time from $a_1, a_2 \xleftarrow{\$} [0, pq)$.

**Encryption.** We encrypt a plaintext integer $m$ as the 2-vector $\boldsymbol{c}$,

$$\boldsymbol{c} = \mathsf{Enc}(m, \mathsf{sk}) = (m + rp)\mathbf{1} + s\boldsymbol{a} \pmod{pq},$$

where $\mathbf{1} = [1\ \ 1]^T$, $r \xleftarrow{\$} [0, q)$, and $s \xleftarrow{\$} [0, pq)$. $r$ and $s$ are independent. We note that two encryptions of the same plaintext are different with very high probability.

**Theorem 3.** *The encryption scheme produces ciphertexts with components which are random integers modulo $pq$.*

Note, however, that the components of the ciphertexts are correlated, and this may be a vulnerability. We discuss this later in this section ("Cryptanalysis").

**Decryption.** To decrypt, we eliminate $s$ from $\boldsymbol{c}$ (modulo $p$), giving

$$\mathsf{Dec}(\boldsymbol{c}, \mathsf{sk}) = \boldsymbol{\gamma}^T \boldsymbol{c} \mod p,$$

where $\boldsymbol{\gamma}^T = (a_2 - a_1)^{-1}[a_2\ \ -a_1]$. We call $\gamma$ the *decryption vector*.

**Addition.** We define the addition operation on ciphertexts as the vector sum modulo $pq$ of the two ciphertext vectors $\boldsymbol{c}$ and $\boldsymbol{c'}$,

$$\mathsf{Add}(\boldsymbol{c}, \boldsymbol{c'}) = \boldsymbol{c} + \boldsymbol{c'} \pmod{pq}.$$

Therefore, if inputs $m, m'$ encrypt as $(m + rp)\mathbf{1} + s\boldsymbol{a}$, $(m' + r'p+)\mathbf{1} + s'\boldsymbol{a}$,

$$\mathsf{Add}(\boldsymbol{c}, \boldsymbol{c'}) = \boldsymbol{c} + \boldsymbol{c'} = (m + m' + (r + r')p)\mathbf{1} + (s + s')\boldsymbol{a}.$$

which is a valid encryption of $m + m'$.

**Multiplication.** If $\boldsymbol{c} = [c_1\ c_2]^T$, we construct the augmented ciphertext vector, $\boldsymbol{c}_\star = [c_1\ c_2\ c_3]^T$, where $c_3 = 2c_1 - c_2$. Thus, $c_3 = (m+rp)+sa_3 \mod pq$, for $a_3 = 2a_1 - a_2$. So,

$$\mathsf{Mult}(\boldsymbol{c}, \boldsymbol{c'}) = \boldsymbol{c} \cdot \boldsymbol{c'} = R(\boldsymbol{c}_\star \circ \boldsymbol{c'}_\star) \pmod{pq},$$

where $\cdot$ is a product on $\mathbb{Z}_{pq}^2$ and $\boldsymbol{c}_\star \circ \boldsymbol{c'}_\star$ is the Hadamard product modulo $pq$ of the two augmented ciphertext vectors $\boldsymbol{c}_\star$ and $\boldsymbol{c'}_\star$.

**Theorem 4.** *If $\boldsymbol{c}$ is an encryption of $m$ and $\boldsymbol{c'}$ is an encryption of $m'$ then $R(\boldsymbol{c}_\star \circ \boldsymbol{c'}_\star) \pmod{pq}$ is an encryption of $mm'$.*

Observe that $\alpha_1, \alpha_2$ in $R$ are public, but give only two equations for the four parameters of the system $a_1, a_2, \sigma, \varrho p$. These equations are quadratic $\mod pq$, and solving them is as hard as semiprime factorisation in the worst case [47].

Also, observe that, independently of $\boldsymbol{a}$,

$$R\boldsymbol{c}_\star = (m + rp)R\mathbf{1}_\star + sR\boldsymbol{a}_\star = (m + rp)\mathbf{1} + s\boldsymbol{a} = \boldsymbol{c},$$

for any ciphertext $\boldsymbol{c}$. Hence re-encrypting a ciphertext gives the identity operation, and discloses no information.

**Hardness.** We can show that this system is at least as hard as SPACDP. In fact,

**Theorem 5.** *SPACDP is of equivalent complexity to the special case of HE2 where $\delta = a_2 - a_1$ ($0 < \delta < q$) is known.*

Without knowing the parameter $\delta = a_2 - a_1$, HE2 cannot be reduced to SPACDP in this way, so HE2 is more secure than HE1.

**Cryptanalysis.** Each new ciphertext $\boldsymbol{c}$ introduces two new unknowns $r, s$ and two equations for $c_1, c_2$. Thus we gain no additional information from a new ciphertext. However, if we can guess, $m$, $m'$ for any two ciphertexts $\boldsymbol{c}, \boldsymbol{c}'$, we can determine

$$(c_1 - m) = rp + sa_1, \qquad (c_2 - m) = rp + sa_2,$$
$$(c_1' - m') = r'p + s'a_1, \qquad (c_2' - m') = r'p + s'a_2,$$
$$\text{so } (c_1 - m)(c_2' - m') - (c_2 - m)(c_1' - m') = (a_2 - a_1)(rs' - r's)p \pmod{pq}$$

Since $a_2 \neq a_1$, and $sr' \neq s'r$ with high probability, this is a nonzero multiple of $p$, $\nu p$ say. We may assume $\nu < q$, so $p = \gcd(\nu p, pq)$. We can now solve the linear system $\gamma^T[\boldsymbol{c}\ \boldsymbol{c}'] = [m\ m'] \mod p$ to recover the decryption vector. This effectively breaks the system, since we can now decrypt an arbitrary ciphertext. We could proceed further, and attempt to infer $a_1$ and $a_2$, but we will not do so.

Note that to break this system, we need to guess two plaintexts, as opposed to one in HE1. The entropy of a pair $(m, m')$ is $2\rho$, so we have effectively squared the number of guesses needed to break the system relative to HE1. So HE2 can tolerate smaller entropy than HE1. We note further that HE2 does not seem immediately vulnerable to known cryptanalytic attacks on HE1 [16, 17, 36].

### 4.2 Insufficient Entropy (HE2N)

In this section we extend HE1N above (section 3.2) to two dimensions.

**Key Generation.** $\mathsf{KeyGen}$ randomly chooses $p, q$ and $\kappa$ according to the bounds given in section 3.2. $\mathbf{1}$ is defined as in section 4.1. $\boldsymbol{a}$, and $R$ are generated as in section 4.1. The secret key is $(\kappa, p, \boldsymbol{a})$, and the public parameters are $pq$ and $R$, defined in section 4.1.

**Encryption.** We encrypt a plaintext integer $m \in [0, M)$ as a 2-vector $\boldsymbol{c}$,

$$\mathsf{Enc}(m, \mathsf{sk}) = \boldsymbol{c} = (m + rp + s\kappa)\mathbf{1} + t\boldsymbol{a} \mod pq,$$

where $r$ is as in section 4.1, $s \xleftarrow{\$} [0, \kappa)$, and $t \xleftarrow{\$} [0, pq)$.

**Decryption.** We decrypt a ciphertext $\boldsymbol{c}$ by $\mathsf{Dec}(\boldsymbol{c}, \mathsf{sk}) = (\boldsymbol{\gamma}^T\boldsymbol{c} \mod p) \mod \kappa$, where $\boldsymbol{\gamma}^T$ is defined as in 4.1.

**Arithmetic.** Addition and multiplication of ciphertexts are as in section 4.1.

**Security.** HE2N has all the properties of HE1N. However, it is more secure, since there is an additional unknown parameter in the ciphertext. We also note that HE2N satisfies Theorem 2, so it inherits the IND-CPA property.

### 4.3 Generalisation of HE2 and HE2N to $k$ Dimensions

The integer vector based approach of HE2 and HE2N can be generalised to vectors of dimension $k$. We do not have space to present this material here, but it may be found in appendix A.

## 5 Experimental Results

HE1, HE1N, HE2, and HE2N have been implemented in pure unoptimised Java using the JScience mathematics library [21]. Secure pseudo-random numbers are generated using the ISAAC algorithm [37], seeded using the Linux `/dev/random` source. This prevents the weakness in ISAAC shown by Aumasson [3].

The evaluation experiment generated 24,000 encrypted inputs and evaluated a polynomial homomorphically on the inputs, using a Hadoop MapReduce (MR) algorithm. On the secure client side, the MR input is generated as pseudo-random $\rho$-bit integers which are encrypted and written to a file with $d$ inputs per line, where $d$ is the degree of the polynomial to be computed. The security parameters $\lambda$ and $\eta$ were selected to be the minimum values required to satisfy the conditions give in sections 3.1, 3.2, 4.1, and 4.2. In addition, the unencrypted result of the computation is computed so that it may checked against the decrypted result of the homomorphic computation. On the Hadoop cluster side, each mapper processes a line of input by homomorphically multiplying together each input on a line and outputs this product. A single reducer homomorphically sums the products. The MR algorithm divides the input file so that each mapper receives an equal number of lines of input, ensuring maximum parallelisation. Finally, on the secure client side, the MR output is decrypted.

Our test environment consisted of a single secure client (an Ubuntu Linux VM with 16GB RAM) and a Hadoop 2.7.3 cluster running in a heterogeneous OpenNebula cloud. The Hadoop cluster consisted of 17 Linux VMs, one master and 16 slaves, each allocated 2GB of RAM. Each experimental configuration of algorithm, polynomial degree ($d$), integer size ($\rho$), and effective entropy of inputs after adding "noise" ($\rho'$, for the 'N' variant algorithms only), was executed 10 times. The means are tabulated in Table 1.

There are some small anomalies in our data. JScience implements arbitrary precision integers as an array of Java `long` (64-bit) integers. This underlying representation may be optimal in some of our test configurations and suboptimal

**Table 1.** Timings for each experimental configuration ($n = 24000$ in all cases, $\lambda > 96$). *Init* is the initialisation time for the encryption algorithm, *Enc* is the mean time to encrypt a single integer, *Exec* is the total MR job execution time, *Prod* is the mean time to homomorphically compute the product of two encrypted integers, *Sum* is the mean time to homomorphically compute the sum of two encrypted integers.

| Alg. | Parameters | | | Encryption | | MR Job | | | Decrypt |
|---|---|---|---|---|---|---|---|---|---|
| | $d$ | $\rho$ | $\rho'$ | Init(s) | Enc($\mu$s) | Exec(s) | Prod($\mu$s) | Sum($\mu$s) | (ms) |
| HE1 | 2 | 32 | n/a | 0.12 | 13.52 | 23.82 | 54.41 | 9.06 | 0.21 |
| HE1 | 2 | 64 | n/a | 0.12 | 16.24 | 23.85 | 60.38 | 8.04 | 0.49 |
| HE1 | 2 | 128 | n/a | 0.15 | 25.73 | 23.77 | 84.69 | 8.43 | 0.28 |
| HE1 | 3 | 32 | n/a | 0.17 | 22.98 | 23.65 | 87.75 | 11.46 | 0.35 |
| HE1 | 3 | 64 | n/a | 0.19 | 34.63 | 24.72 | 95.68 | 12.37 | 0.45 |
| HE1 | 3 | 128 | n/a | 0.42 | 54.83 | 26.05 | 196.71 | 14.07 | 0.55 |
| HE1 | 4 | 32 | n/a | 0.28 | 43.36 | 24.48 | 108.72 | 13.75 | 0.5 |
| HE1 | 4 | 64 | n/a | 0.53 | 58.85 | 26.41 | 227.44 | 15.85 | 3.59 |
| HE1 | 4 | 128 | n/a | 1.36 | 104.95 | 28.33 | 484.95 | 16.92 | 5.67 |
| HE1N | 2 | 1 | 32 | 0.22 | 32.99 | 22.94 | 88.38 | 8.53 | 3.35 |
| HE1N | 2 | 1 | 64 | 0.39 | 52.63 | 26.24 | 168.54 | 12.39 | 3.56 |
| HE1N | 2 | 1 | 128 | 1.2 | 89.01 | 26.18 | 226.2 | 13.16 | 8.1 |
| HE1N | 2 | 8 | 32 | 0.6 | 57.88 | 25.9 | 177.36 | 11.17 | 7.18 |
| HE1N | 2 | 8 | 64 | 0.32 | 43.93 | 26.53 | 96.78 | 12.18 | 2.27 |
| HE1N | 2 | 8 | 128 | 1.13 | 78.11 | 24.42 | 212.75 | 11.07 | 8.4 |
| HE1N | 2 | 16 | 64 | 0.33 | 53.97 | 27.15 | 168 | 13.67 | 4.47 |
| HE1N | 2 | 16 | 128 | 0.63 | 68.73 | 25.22 | 194.42 | 11.01 | 7.65 |
| HE1N | 3 | 1 | 32 | 8.54 | 183.19 | 24.24 | 522.07 | 12.06 | 9.09 |
| HE1N | 3 | 1 | 64 | 3.67 | 125 | 29.49 | 467.36 | 18.22 | 11.43 |
| HE1N | 3 | 1 | 128 | 27.84 | 313.76 | 26.94 | 1235.77 | 15.04 | 11.75 |
| HE1N | 3 | 8 | 32 | 115 | 462.45 | 32.61 | 1556.17 | 21.11 | 19.79 |
| HE1N | 3 | 8 | 64 | 9.75 | 180.08 | 25.87 | 500.62 | 15.03 | 10.39 |
| HE1N | 3 | 8 | 128 | 36.05 | 259.15 | 30.1 | 836.27 | 20.68 | 11.45 |
| HE1N | 3 | 16 | 64 | 30.96 | 378.99 | 28.24 | 1338.33 | 15.51 | 13.3 |
| HE1N | 3 | 16 | 128 | 8.13 | 226.32 | 27.92 | 621.95 | 18.01 | 10.89 |
| HE2 | 2 | 32 | n/a | 0.16 | 85.79 | 26.82 | 305.52 | 11.68 | 4.83 |
| HE2 | 2 | 64 | n/a | 0.17 | 95.92 | 29.71 | 354.79 | 16.9 | 3.26 |
| HE2 | 2 | 128 | n/a | 0.22 | 132.53 | 32.84 | 540.78 | 22.83 | 4.92 |
| HE2 | 3 | 32 | n/a | 0.23 | 130.3 | 31.18 | 513.93 | 23.77 | 6.52 |
| HE2 | 3 | 64 | n/a | 0.29 | 145.62 | 32.84 | 615.9 | 24.61 | 6.3 |
| HE2 | 3 | 128 | n/a | 0.52 | 249.47 | 29.54 | 1443.82 | 16.56 | 18.34 |
| HE2 | 4 | 32 | n/a | 0.39 | 175.63 | 29.5 | 733.23 | 20.69 | 6.01 |
| HE2 | 4 | 64 | n/a | 0.7 | 255.3 | 29.55 | 1578.39 | 18.29 | 16.24 |
| HE2 | 4 | 128 | n/a | 2.7 | 465.51 | 37.47 | 2943.91 | 22.15 | 15.41 |
| HE2N | 2 | 1 | 32 | 0.27 | 147.83 | 29.74 | 571.94 | 16.58 | 5.66 |
| HE2N | 2 | 1 | 64 | 0.43 | 202.74 | 33.36 | 1291.68 | 18.3 | 13.23 |
| HE2N | 2 | 1 | 128 | 1.58 | 354.19 | 33.76 | 1977.51 | 17.13 | 12.46 |
| HE2N | 2 | 8 | 32 | 0.59 | 234.83 | 31.42 | 1413.31 | 15.21 | 14.92 |
| HE2N | 2 | 8 | 64 | 0.33 | 163.78 | 27.42 | 635.64 | 13.6 | 6.18 |
| HE2N | 2 | 8 | 128 | 0.9 | 307.68 | 36.32 | 1850.83 | 21.71 | 15.79 |
| HE2N | 2 | 16 | 64 | 0.42 | 208.1 | 29.96 | 1230.56 | 13.41 | 13.16 |
| HE2N | 2 | 16 | 128 | 0.73 | 274.48 | 30.82 | 1585.1 | 14.85 | 15.04 |
| HE2N | 3 | 1 | 32 | 5.72 | 651.1 | 36.49 | 3438.96 | 18.67 | 19.05 |
| HE2N | 3 | 1 | 64 | 4.45 | 477.52 | 35.33 | 3073.46 | 18.75 | 19.77 |
| HE2N | 3 | 1 | 128 | 26.83 | 1192.79 | 43.23 | 6416.43 | 22.48 | 25.12 |
| HE2N | 3 | 8 | 32 | 87.38 | 1658.36 | 49.63 | 8139.19 | 23.71 | 27.24 |
| HE2N | 3 | 8 | 64 | 5.21 | 607.75 | 36.54 | 3337.1 | 22.28 | 17.39 |
| HE2N | 3 | 8 | 128 | 17.14 | 945.64 | 40.49 | 4620.69 | 25.91 | 22.41 |
| HE2N | 3 | 16 | 64 | 39.19 | 1368.18 | 44.88 | 7005.7 | 24.1 | 28.3 |
| HE2N | 3 | 16 | 128 | 11.39 | 774.07 | 36.05 | 3845.1 | 20.29 | 20.74 |

in others, causing anomalous results. Another possibility is that the unexpected results are due to garbage collection in the JVM heap, which may be more prevalent in certain test configurations.

We may compare these results with those reported in the literature. Our results compare extremely favourably with Table 2 of [42]. For encryption, our results are, in the best case, 1000 times faster than those presented there, and, in the worst case, 10 times faster. For decryption, our results are comparable. However, it should be noted that to decrypt our results we take the moduli for large primes rather than 2 as in [42], which is obviously less efficient. For homomorphic sums and products, our algorithms perform approximately 100 times faster. [42] only provides experimental data for computing degree 2 polynomials. We provide experimental results for higher degree polynomials.

Similarly, compared with Fig. 13 of Popa et al. [46], our encryption times for a 32-bit integer are considerably faster. While a time for computing a homomorphic sum on a column is given in Fig. 12, it is unclear how many rows exist in their test database. Nevertheless, our results for computing homomorphic sums compare favourably with those given. Since CryptDB [46] only supports homomorphic sums and cannot compute an inner product, we can only compare the homomorphic sum timings.

Table 1 of [52] is unclear whether the timings are aggregate or per operation. Even assuming that they are aggregate, our results are approximately 100 times faster for homomorphic sum and product operations. Crypsis [52] uses two different encryption schemes for integers, ElGamal [26] and Paillier [45], which only support addition or multiplication but not both. No discussion of computation of an inner product is made in [52] but we expect that the timings would be considerably worse as data encrypted using ElGamal to compute the products would have to be shipped back to the secure client to be re-encrypted using Paillier so that the final inner product could be computed.

Varia et al. [55] present experimental results of applying their HETest framework to HELib [35]. Varia et al. show timings $10^4$ to $10^6$ times slower than that of computations on unencrypted data. Although it is unclear exactly which circuits are being computed, the timings given are in seconds, so we believe that HELib will not be a serious candidate for SSCC in the immediate future.

As reported in [24], the current performance of FHEW [25] is poor compared with unencrypted operations. The authors report that FHEW processed a single homomorphic NAND operation followed by a re-encryption in 0.69s and using 2.2GB of RAM. Therefore, we also believe that FHEW is not a candidate for SCCC, as it currently stands.

Although claims regarding its performance have been made in the press [54], no benchmarking statistics have been made publicly available for Mi-

crosoft's SEAL library [41]. However, in [2], it is reported that, for SEAL v1, the time to perform one multiplication is approximately 140ms.

With regard to FV-NFLib [20], Bonte et al. [10] recently reported a significant decrease in the time to evaluate a four layer *Group Method of Data Handling* (GMDH) neural network [11] from 32s to 2.5s, as a result of their novel encoding of the inputs.

Aguilar-Melchor et al [2] report their experimental findings regarding HELib-MP [48]. They show that HELib-MP outperforms FV-NFLib for large (2048-bit) plaintexts. They further go on to benchmark HELib-MP by computing RSA-2048 and ECC-ElGamal-P256. An exponentiation in RSA-2048 takes between 157ms and 1.8s depending on the window size and number of multiplications required. For ECC-ElGamal-P256, an elliptic curve multiplication takes between 96ms and 242ms depending on window size and number of elliptic curve additions.

Catalano et al. [15] provide experimental results for their work. For 128-bit plaintexts, our algorithms are approximately 10 to 1000 times faster at performing a multiplication operation and our most complex algorithm, HE2N, is roughly equal to their fastest, an extension of Joye-Libert [38], for additions.

Yu et al. [58] give experimental results for their implementation of the Zhou and Wornell scheme [59]. From their Figures 3 to 5, it is hard to compare our scheme with theirs directly but it would appear that our vector based schemes are at least comparable in performance to theirs.

## 6 Conclusion

In this paper we have presented several new homomorphic encryption schemes intended for use in a practical SSCC system. We envisage that the majority of computation on integer big data, outside of scientific computing, will be computing low degree polynomials on integers, or fixed-point decimals which can be converted to integers. Our somewhat homomorphic schemes are perfectly suited to these types of computation.

Our evaluation has only concerned one- or two-dimensional ciphertexts and polynomials of degree up to four. We intend to investigate higher degree polynomials in future work. We believe that HE1N and HE2N provide strong security, even for low-entropy data, as they satisfy the desirable IND-CPA property. If a user has a high confidence in the entropy of the input data, HE2 may provide sufficient security.

As they are only somewhat homomorphic, each of these schemes require that the computational result cannot grow bigger than the secret modulus. In the case of the "noise" variants, we also have to consider the noise term growing large. So, as they stand, these schemes can only compute polynomials of suit-

ably bounded degree. However, we believe this is adequate for most practical purposes.

The schemes presented in sections 3 and 4 extend to a hierarchy of systems, HE$k$, with increasing levels of security. These are presented in appendix A and may be investigated further in future work.

We have implemented and evaluated the HE1, HE1N, HE2 and HE2N schemes as part of an SSCC system as discussed in section 5. Our results are extremely favourable by comparison with existing methods. In some cases, they outperform those methods by a factor of 1000. This clearly demonstrates the practical applicability of our schemes. Furthermore, our MapReduce job execution times remain low even when using the largest set of parameters for HE2N. We believe that this demonstrates the advantages of our schemes for encrypted computations on fixed-point data in the cloud.

## A Generalisation to $k$ Dimensions

In this appendix, we generalise HE2 and HE2N to $k$-vectors. HE1 and HE1N are the cases for $k = 1$ and HE2 and HE2N are the cases for $k = 2$.

### A.1 Sufficient Entropy (HE$k$)

We generalise HE2 to $k$ dimensions.

**Key Generation.** $\mathsf{KeyGen}$ randomly chooses $p$ and $q$ according to the bounds given in section 4.1. $\mathsf{KeyGen}$ sets $\boldsymbol{a}_j \xleftarrow{\$} [1, pq)^k$, $\forall j \in [1, k-1]$, and $R$ (detailed in "Multiplication" below). The secret key $\mathsf{sk}$ is $(p, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1})$, and the public parameters are $pq$ and $R$.

**Computational Overhead.** The computational overhead increases, the number of arithmetic operations per plaintext multiplication is $O(k^3)$, and the space requirement per ciphertext is $O(k)$, by comparison with HE1.

**Encryption.** A plaintext, $m \in [0, M]$, is enciphered as

$$\mathsf{Enc}(m, \mathsf{sk}) = \boldsymbol{c} = (m + rp)\mathbf{1} + \textstyle\sum_{j=1}^{k-1} s_j \boldsymbol{a}_j \mod pq$$

where $\boldsymbol{c}$ is a $k$-vector, $r \xleftarrow{\$} [0, q)$, and $\forall j, s_j \xleftarrow{\$} [0, pq)$. Let $\boldsymbol{a}_0 = \mathbf{1}$, and $A_k = [\boldsymbol{a}_0\ \boldsymbol{a}_1\ \ldots\ \boldsymbol{a}_{k-1}]$. We wish the columns of $A_k$ to be a basis for $\mathbb{Z}_{pq}^k$. We can show that they do so with high probability. If they do not, we generate new vectors until they do.

**Lemma 2.** $\Pr(\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}$ *do not form a basis*$) \leq (k-1)(1/p + 1/q)$.

We extend our definition of an augmented vector $\boldsymbol{v}_\star$, for a $k$-vector, $\boldsymbol{v}$, such that $\boldsymbol{v}_\star$ is a $\binom{k+1}{2}$-vector, with components $v_i$ $(1 \leq i \leq k)$ followed by $2v_i - v_j$ $(1 \leq i < j \leq k)$. In general, for $\ell > k$, $v_\ell = 2v_i - v_j$, where $\ell = \binom{i}{2} + k + j - 1$.

Note that $\boldsymbol{v}_\star = U_k \boldsymbol{v}$ for a $\binom{k+1}{2} \times k$ matrix with entries $0, \pm 1, 2$, and whose first $k$ rows form the $k \times k$ identity matrix $I_k$. Note that $\boldsymbol{v}_\star = U_k \boldsymbol{v}$ implies that $\mathbf{1}_\star$ is the $\binom{k+1}{2}$ vector of 1's, and that $*$ is a linear mapping, i.e. $(r_1 \boldsymbol{v}_1 + r_2 \boldsymbol{v}_2)_\star = r_1 \boldsymbol{v}_{1*} + r_2 \boldsymbol{v}_{2*}$.

**Decryption.** $\mathsf{Dec}(\boldsymbol{c}, \mathsf{sk}) = \boldsymbol{\gamma}^T \boldsymbol{c} \mod p$, where $\boldsymbol{\gamma}^T = (A_k^{-1})_1$ is the first row of $A_k^{-1}$. We call $\boldsymbol{\gamma}$ the *decryption vector*, as in HE2.

**Addition.** Addition is the vector sum of the ciphertext vectors as in HE2.

**Multiplication.** Consider the Hadamard product of two augmented ciphertext vectors, $\boldsymbol{c}_\star \circ \boldsymbol{c}'_\star$. For notational brevity, let $\tilde{m} = m + rp$.

$$\begin{aligned} \boldsymbol{c}_\star \circ \boldsymbol{c}'_\star &= \big(\tilde{m}\mathbf{1}_\star + \textstyle\sum_{j=1}^{k-1} s_j \boldsymbol{a}_{\star j}\big) \circ \big(\tilde{m}'\mathbf{1}_\star + \textstyle\sum_{j=1}^{k-1} s'_j \boldsymbol{a}_{\star j}\big) \\ &= \tilde{m}\tilde{m}'\mathbf{1}_\star + \textstyle\sum_{j=1}^{k-1}(\tilde{m}s'_j + \tilde{m}'s_j)\boldsymbol{a}_{\star j} + \sum_{j=1}^{k-1} s_j s'_j \boldsymbol{a}_{\star j} \circ \boldsymbol{a}_{\star j} \\ &\qquad + \textstyle\sum_{1 \le i < j \le k-1}(s_i s'_j + s'_i s_j)\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}, \end{aligned}$$

since $\mathbf{1}_\star \circ \boldsymbol{v}_\star = \boldsymbol{v}_\star$ for any $\boldsymbol{v}$. There are $\binom{k}{2}$ product vectors, which we must eliminate using the re-encryption matrix $R$, a $k \times \binom{k+1}{2}$.

**Lemma 3.** *Let $A_{\star k} = [\boldsymbol{a}_{\star 0}\ \boldsymbol{a}_{\star 1}\ \ldots\ \boldsymbol{a}_{\star,k-1}]$, where the columns of $A_k$ form a basis for $\mathbb{Z}_{pq}^k$. If $RA_{\star k} = A_k$, then $R\boldsymbol{v}_\star = \boldsymbol{v}$ for all $\boldsymbol{v} \in \mathbb{Z}_{pq}^k$.*

The condition $RA_{\star k} = A_k$ can be written more simply, since it is $RU_k A_k = A_k$. Postmultiplying by $A_k^{-1}$ gives $RU_k = I_k$. Now, since $RA_{\star k} = A_k$, we have

$$R(\boldsymbol{c}_\star \circ \boldsymbol{c}'_\star) \;=\; (mm' + \hat{r}p)\mathbf{1} + \textstyle\sum_{j=1}^{k-1} \hat{s}_j \boldsymbol{a}_j + \sum_{1 \le i \le j \le k-1} \hat{s}_{ij} R(\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}),$$

where $\hat{r}$, $\hat{s}_j$ and $\hat{s}_{ij}$ $(1 \le i < j \le k-1)$ are some integers.

There are $k(\binom{k+1}{2} - k) = k\binom{k}{2}$ undetermined parameters $R_{i\ell}$, $1 \le i \le k$, $k < \ell \le \binom{k+1}{2}$. We now determine these by setting

$$R(\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}) \;=\; \varrho_{ij} p\mathbf{1} + \textstyle\sum_{l=1}^{k-1} \sigma_{ijl} \boldsymbol{a}_l \tag{2}$$

Thus we have $k\binom{k}{2}$ new unknowns, the $\varrho$'s and $\sigma$'s, and $k\binom{k}{2}$ linear equations for the $k\binom{k}{2}$ unassigned $R_{i\ell}$'s. Let $A_{\star k}^{\circ 2}$ be the $\binom{k+1}{2} \times \binom{k+1}{2}$ matrix with columns $\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}$ $(0 \le i < j < k)$, and let $C_k$ be the $k \times \binom{k}{2}$ matrix with columns $\varrho_{ij} p\mathbf{1} + \sum_{l=1}^{k-1} \sigma_{ijl} \boldsymbol{a}_l$ $(0 < i < j < k)$. Then the equations for the $R_{i\ell}$ can be written as

$$RA_{\star k}^{\circ 2} \;=\; [A_k \mid C_k]\,. \tag{3}$$

giving $k\binom{k+1}{2}$ linear equations for the $k\binom{k+1}{2}$ $R_{i\ell}$'s in terms of quadratic functions of the $k(k-1)$ $a_{ij}$'s $(1 \le i \le k, 1 \le j \le k-1)$, which are undetermined. Thus the system has $k(k-1)$ parameters that cannot be deduced from $R$.

The system of equations (3) has a solution provided that $A_{\star k}^{\circ 2}$ has an inverse $\bmod pq$. We prove that this is true with high probability. Again, in the unlikely event that this is not true, we generate new vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}$ until it is.

**Theorem 6.** *$A^{\circ 2}_{\star k}$ has no inverse* $\bmod pq$ *with probability at most* $(k^2-1)(1/p+1/q)$.

Note that Theorem 6 subsumes Lemma 2, since the first $k$ columns of $A^{\circ 2}_{\star k}$ contain $A_k$ as a submatrix, and must be linearly independent.

Each $\boldsymbol{c}$ introduces $k$ new parameters $rp, s_1, \ldots, s_{k-1}$ and $k$ equations, so the number of undetermined parameters is always $k(k-1)$.

**Cryptanalysis.** Note that $p$ can be determined from $m_i$ for $k$ ciphertexts. Let

$$C = [\boldsymbol{c}_1 - m_1\mathbf{1} \ \ldots \ \boldsymbol{c}_k - m_k\mathbf{1}], \quad A_k = [\mathbf{1} \ \boldsymbol{a}_1 \ \ldots \ \boldsymbol{a}_{k-1}]$$

and let

$$W = \begin{bmatrix} r_1p & r_2p & \ldots & r_kp \\ s_{1,1} & s_{2,1} & \ldots & s_{k,1} \\ \vdots & & & \vdots \\ s_{1,k-1} & s_{2,k-1} & \ldots & s_{k,k-1} \end{bmatrix}, \quad W' = \begin{bmatrix} r_1 & r_2 & \ldots & r_k \\ s_{1,1} & s_{2,1} & \ldots & s_{k,1} \\ \vdots & & & \vdots \\ s_{1,k-1} & s_{2,k-1} & \ldots & s_{k,k-1} \end{bmatrix},$$

where $r_i, s_{ij}$ refer to $\boldsymbol{c}_i$. Then $C = A_kW$, and so $\det C = \det A_k \det W$. Note that $\det W = p \det W'$, so $\det C$ is a multiple of $p$. Now $\det C$ can be determined in $O(k^3)$ time and, if it is nonzero, $p$ can be determined as $\gcd(\det C, pq)$.

**Lemma 4.** $\Pr(\det C = 0 \bmod pq) \le (2k-1)(1/p + 1/q)$.

Once we have recovered $p$, we can use the known $m_i$ to determine the decryption vector $\boldsymbol{\gamma}$, by solving linear equations. Let $C_0 = [\boldsymbol{c}_1 \ \boldsymbol{c}_2 \ \ldots \ \boldsymbol{c}_k]$, $\boldsymbol{m}^T = [m_1 \ m_2 \ \ldots \ m_k]$.

**Lemma 5.** $\Pr(\det C_0 = 0 \bmod pq) \le (2k-1)(1/p + 1/q)$.

Thus, with high probability, we can uniquely solve the system $\boldsymbol{\gamma}^T C_0 = \boldsymbol{m}^T \bmod p$, to recover $\boldsymbol{\gamma}$ and enable decryption of an arbitrary ciphertext. However, encryption of messages is not possible, since we gain little information about $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$. Note also that, if we determined $p$ by some means other than using $k$ known plaintexts, it is not clear how to recover $\boldsymbol{\gamma}$.

To break this system, we need to guess $k$ plaintexts. The entropy of a $k$-tuple of plaintexts $(m_1, m_2, \ldots, m_k)$ is $k\rho$, so effectively we need $\mu^k$ guesses, where $\mu$ is the number of guesses needed to break HE1. So HE$k$ can tolerate much smaller entropy than HE1, provided $k$ is large enough. If $k$ is sufficiently large, the scheme appears secure without adding noise, but does not have the other advantages of adding noise.

**Fixing an Insecurity for $k > 2$** The decryption vector for HE$k$ is $\boldsymbol{\gamma}^T = (A_k^{-1})_1$. Note that $\boldsymbol{\gamma}^T\mathbf{1} = 1$ and $\boldsymbol{\gamma}^T\boldsymbol{a}_i = 0$ ($i \in [1, k-1]$), since $\boldsymbol{\gamma}^T\boldsymbol{a}_i = I_{1i}$

($i \in [0, k-1]$).

The equations $$R(\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}) \;=\; p\varrho_{ij}\,\mathbf{1} + \textstyle\sum_{l=1}^{k-1} \sigma_{ijl}\,\boldsymbol{a}_l, \tag{4}$$

define a product $\cdot$ on $\mathbb{Z}_{pq}^k$ so that $\boldsymbol{c} \cdot \boldsymbol{c}' = R(\boldsymbol{c}_\star \circ \boldsymbol{c}'_\star)$. This product is linear, commutative and distributive, since $R$ and $\star$ are linear operators, and $\circ$ is commutative and distributive. So we have an algebra $\mathcal{A}_k$, with unit element $\mathbf{1}$ [51]. The $\varrho_{ij}, \sigma_{ijl}$ ($i, j, l \in [1, k-1]$) are the *structure constants* of the algebra. In general, $\mathcal{A}_k$ will not be associative, i.e. we can have $(\boldsymbol{c}_1 \cdot \boldsymbol{c}_2) \cdot \boldsymbol{c}_3 \neq \boldsymbol{c}_1 \cdot (\boldsymbol{c}_2 \cdot \boldsymbol{c}_3)$ This leads to a potential insecurity. We must have

$$\boldsymbol{\gamma}^T((\boldsymbol{c}_1 \cdot \boldsymbol{c}_2) \cdot \boldsymbol{c}_3) \;=\; \boldsymbol{\gamma}^T(\boldsymbol{c}_1 \cdot (\boldsymbol{c}_2 \cdot \boldsymbol{c}_3)) \qquad (\mathrm{mod}\ p), \tag{5}$$

in order to have correct decryption. The *associator* for $\mathcal{A}_k$ is

$$[\boldsymbol{c}_i, \boldsymbol{c}_j, \boldsymbol{c}_l] \;= \boldsymbol{c}_i \cdot (\boldsymbol{c}_j \cdot \boldsymbol{c}_l) - (\boldsymbol{c}_i \cdot \boldsymbol{c}_j) \cdot \boldsymbol{c}_l = rp\mathbf{1} + \textstyle\sum_{l=1}^{k-1} s_l\,\boldsymbol{c}_l \qquad (\mathrm{mod}\ pq).$$

Thus $[\boldsymbol{c}_i, \boldsymbol{c}_j, \boldsymbol{c}_l]$ is an encryption of 0. If we can find $k$ associators from $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ which violate (5), with high probability we have $k$ linearly independent associators. We can use these to make a collision attack on HE$k$, similar to that described in section 3.1. We use the $\gcd$ method to determine $p$, and then $\boldsymbol{\gamma}$, as described in section A.1. In fact all we need is that (5) holds for any associator. That is, for all $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$, we need

$$\boldsymbol{\gamma}^T((\boldsymbol{c}_1 \cdot \boldsymbol{c}_2) \cdot \boldsymbol{c}_3) \;=\; \boldsymbol{\gamma}^T(\boldsymbol{c}_1(\cdot\boldsymbol{c}_2 \cdot \boldsymbol{c}_3)) \qquad (\mathrm{mod}\ pq),$$

or, equivalently, using the Chinese Remainder Theorem,

$$\boldsymbol{\gamma}^T((\boldsymbol{c}_1 \cdot \boldsymbol{c}_2) \cdot \boldsymbol{c}_3) \;=\; \boldsymbol{\gamma}^T(\boldsymbol{c}_1 \cdot (\boldsymbol{c}_2 \cdot \boldsymbol{c}_3)) \qquad (\mathrm{mod}\ q). \tag{6}$$

By linearity, (6) holds if and only if it holds for all basis elements, excluding the identity. That is, for all $i, j, l \in [1, k-1]$, we need

$$\boldsymbol{\gamma}^T(\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l)) \;=\; \boldsymbol{\gamma}^T((\boldsymbol{a}_i \cdot \boldsymbol{a}_j) \cdot \boldsymbol{a}_l) \qquad (\mathrm{mod}\ q). \tag{7}$$

The associator for $\mathcal{A}_k$ is

$$[\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{a}_l] \;= \boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l) - (\boldsymbol{a}_i \cdot \boldsymbol{a}_j) \cdot \boldsymbol{a}_l = rp\mathbf{1} + \textstyle\sum_{l=1}^{k-1} s_l\,\boldsymbol{a}_l \qquad (\mathrm{mod}\ pq),$$

for some integers $r, s_1, \ldots, s_{k-1}$, and so $\boldsymbol{\gamma}^T[\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{a}_l] = rp$.

If $\mathcal{A}_k$ is associative, the problem does not arise, since (7) will be satisfied automatically. Associativity holds if $k \leq 2$. All we have to check is that $\boldsymbol{a} \cdot (\boldsymbol{a} \cdot \boldsymbol{a}) = (\boldsymbol{a} \cdot \boldsymbol{a}) \cdot \boldsymbol{a}$, which is true by commutativity. Thus HE1, HE2 cannot be attacked in this way.

Requiring associativity in $\mathcal{A}_k$ overconstrains the system, imposing $k\binom{k+1}{2}$ equations on the $k\binom{k+1}{2}$ structure constants. With only $k(k-1)$ undetermined parameters, this is too much. But all we need is that (7) holds. We have

**Lemma 6.** *Equation* (7) *holds if and only if* $\sum_{t=1}^{k-1} \sigma_{jlt}\varrho_{it} \;=\; \sum_{t=1}^{k-1} \sigma_{ijt}\varrho_{lt}$ $(\mathrm{mod}\ q)$, $\forall i, j, l \in [1, k-1]$.

Now we can ensure (7) by giving the $\varrho_{ij}$ a multiplicative structure.

**Lemma 7.** *Let* $\tau, \varrho_i \xleftarrow{\$} [0, q)$ $(i \in [1, k-1])$*, let* $\varrho_{ij} = \varrho_i \varrho_j \mod q$*, and let the* $\sigma_{ijl}$ *satisfy* $\sum_{l=1}^{k-1} \sigma_{ijl} \varrho_l = \tau \varrho_i \varrho_j \pmod q$ *for all* $i, j \in [1, k-1]$*. Then, for all* $i, j, \ell \in [1, k-1]$*,* $\boldsymbol{\gamma}^T(\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l)) = \tau \varrho_i \varrho_j \varrho_l \mod q$*, the symmetry of which implies* (7)*.*

Thus the conditions of Lemma 7 are sufficient to remove the insecurity. The price is that we now have $(k-1)\binom{k}{2} + (k-1) + k(k-1) = (k+1)\binom{k}{2} + k - 1$ parameters and $k\binom{k}{2}$ equations. There are $\binom{k}{2} + (k-1) = (k+2)(k-1)/2$ independent parameters. This is fewer than the original $k(k-1)$, but remains $\Omega(k^2)$.

### A.2 Insufficient Entropy (HE$k$N)

We generalise HE2N to $k$ dimensions.

**Key Generation.** KeyGen, randomly chooses $\kappa$, $p$ and $q$ as outlined in section 4.2, and sets $\boldsymbol{a}_j$ $\forall j$ and $R$ as in A.1. The secret key, sk, is ($\kappa$, $p$, $\boldsymbol{a}_1$, …, $\boldsymbol{a}_{k-1}$), and the public parameters are $pq$ and $R$. Note that, as a result of adding the "noise" term, defence against non-associativity is not required.

**Encryption.** A plaintext, $m \in [0, M]$, is enciphered as

$$\mathsf{Enc}(m, \mathsf{sk}) = \boldsymbol{c} = (m + rp + s\kappa)\mathbf{1} + \textstyle\sum_{j=1}^{k-1} t_j \boldsymbol{a}_j \pmod{pq}$$

where $r, s$ are as in section 4.2, and $t_j \xleftarrow{\$} [0, pq)$ $\forall j \in [1, k)$.

**Decryption.** If $\boldsymbol{\gamma}^T$ is defined as in section A.1, a ciphertext is deciphered by,

$$\mathsf{Dec}(\boldsymbol{c}, \mathsf{sk}) = (\boldsymbol{\gamma}^T \boldsymbol{c} \mod p) \mod \kappa.$$

**Arithmetic.** Addition and multiplication of ciphertexts are as in section A.1.

**Security.** The effective entropy of HE$k$N is $\rho' = k(\rho + \lg \kappa)$. Thus, as we increase $k$, the "noise" term can be made smaller while still providing the requisite level of entropy.

Clearly HE$k$N also inherits the conclusions of Theorem 2, so this system also satisfies IND-CPA.

## B Proofs

**Theorem 1.** *An attack against HE1 is successful in polynomial time if and only if we can factorise a distinct semi-prime in polynomial time.*

*Proof.* Suppose that we have an unknown plaintext $m$, encrypted as $c = m + rp \mod pq$, where $r \xleftarrow{\$} [1, q)$.

If we can factor $pq$ in polynomial time, we can determine $p$ and $q$ in polynomial time, since we know $p < q$. Therefore, we can determine $m = c \bmod p$.

If we can determine $m$ given $c$ for arbitrary $m$, then we can determine $rp = c - m$. We are given $qp$, and we know $0 < r < q$, so $\gcd(rp, qp)$ must be $p$, and we can compute $p$ in polynomial time. Now, given $p$, we can determine $q$ as $qp/p$. Hence, we can factorise $pq$ in polynomial time. □

**Lemma 1.** *If the inputs $m$ have entropy $\rho$ then, for any two independent inputs $m_1, m_2$, $\Pr(m_1 = m_2) \leq 2^{-\rho}$.*

*Proof.* $\Pr(m_1 = m_2) = \sum_{i=0}^{M-1} \xi_i^2 = 2^{-H_2} \leq 2^{-\rho}$, since $H_2 \geq H_\infty = \rho$. □

**Theorem 2.** *For any encryption $c$, $c \bmod \kappa$ is polynomial time indistinguishable from the uniform distribution on $[0, \kappa)$. Thus HE1N satisfies IND-CPA, under the assumption that SPACDP is not polynomial time solvable.*

*Proof.*

$$c = m + s\kappa + rp = m + rp \mod \kappa,$$

where $r \xleftarrow{\$} [1, q)$. Thus, for $i \in [0, \kappa)$,

$$\begin{aligned}\Pr\left(c \bmod \kappa = i\right) &= \Pr(m + rp = i \mod \kappa) \\ &= \Pr\left(r = p^{-1}(i - m) \mod \kappa\right) \\ &\in \left\{\lfloor q/\kappa \rfloor 1/q, \lceil q/\kappa \rceil 1/q\right\} \\ &\in [1/\kappa - 1/q, 1/\kappa + 1/q],\end{aligned}$$

where the inverse $p^{-1}$ of $p$ mod $\kappa$ exists since $p$ is a prime. Hence the total variation distance from the uniform distribution is

$$\tfrac{1}{2} \sum_{i=0}^{\kappa-1} |\Pr\left(c \bmod \kappa = i\right) - 1/\kappa| < \kappa/q.$$

This is exponentially small in the security parameter $\lambda$ of the system, so the distribution of $c \bmod \kappa$ cannot be distinguished in polynomial time from the uniform distribution. Note further that $c_1 \bmod \kappa$, $c_2 \bmod \kappa$ are independent for any two ciphertexts $c_i = m_i + s_i\kappa + r_i p$ $(i = 1, 2)$, since $r_1, r_2$ are independent.

To show IND-CPA, suppose now that known plaintexts $\mu_1, \ldots, \mu_n$ are encrypted by an oracle for HE1N, giving ciphertexts $c_1, \ldots, c_n$. Then, for $r_i \xleftarrow{\$} [0, q)$, $s_i \xleftarrow{\$} [0, \kappa)$, we have an SPACDP with ciphertexts $c_i = m_i + s_i\kappa + r_i p$, and the approximate divisor $p$ cannot be determined in polynomial time in the worst case. However, the offsets in this SPACDP are all of the form $\mu_i + s_i\kappa$, for known $m_i$, and we must make sure this does not provide information about $p$. To show this, we rewrite the SPACDP as

$$c_i = \mu_i + s_i\kappa + r_i p = \mu_i' + s_i'\kappa, \quad (i = 1, 2, \ldots, n), \tag{8}$$

where $s'_i = s_i + \lfloor (m_i + r_i p)/\kappa \rfloor$, and $\mu'_i = \mu_i + r_i p \pmod{\kappa}$. Now we may view (8) as an ACDP, with "encryptions" $\mu'_i$ of the $\mu_i$, and approximate divisor $\kappa$. Since ACDP is at least as hard as SPACDP, and the offsets $\mu'_i$ are polynomial time indistinguishable from uniform $[0, \kappa)$, from above, we will not be able to determine $\kappa$ in polynomial time. Now, the offsets $m'_1, m'_2$ of any two plaintexts $m_1, m_2$ are polynomial time indistinguishable from $m'_2, m'_1$, since they are indistinguishable from two independent samples from uniform $[0, \kappa)$. Therefore, in polynomial time, we will not be able to distinguish between the encryption $c_1$ of $m_1$ and the encryption $c_2$ of $m_2$. □

**Theorem 3.** *The encryption scheme produces ciphertexts with components which are random integers modulo $pq$.*

*Proof.* Consider a ciphertext vector which encrypts the plaintext, $m$, and the expression $m + rp + sa \mod pq$ which represents one of its elements. Then $r \xleftarrow{\$} [0, q)$, $s \xleftarrow{\$} [0, pq)$.

Consider first $m + sa$. We know that $a^{-1} \mod pq$ exists because $a \neq 0 \pmod p$ and $\mod q$). Thus, conditional on $r$,

$$\Pr[m + rp + sa = i \bmod pq] = \Pr[s = a^{-1}(i - m - rp) \bmod pq] = \frac{1}{pq}.$$

Since this holds for any $i \in [0, pq)$, $m + ra + sp \mod pq$ is a uniformly random integer from $[0, pq)$. □

**Theorem 4.** *If $\boldsymbol{c}$ is an encryption of $m$ and $\boldsymbol{c'}$ is an encryption of $m'$ then $R(\boldsymbol{c}_\star \circ \boldsymbol{c}'_\star) \pmod{pq}$ is an encryption of $mm'$.*

*Proof.* Consider the Hadamard product modulo $pq$, $\boldsymbol{c}_\star \circ \boldsymbol{c}'_\star$, of the two augmented ciphertext vectors $\boldsymbol{c}_\star$ and $\boldsymbol{c}_\star{}'$:

$$\boldsymbol{z}_\star = \boldsymbol{c}_\star \circ \boldsymbol{c}'_\star = \begin{bmatrix} c_1 c'_1 \\ c_2 c'_2 \\ c_3 c'_3 \end{bmatrix} \mod pq$$

Therefore, if inputs $m, m'$ are encrypted as $(m + rp)\mathbf{1} + s\boldsymbol{a}$, $(m' + r'p)\mathbf{1} + s'\boldsymbol{a}$, we first calculate

$$\begin{aligned} \boldsymbol{z}_\star &= (m + rp)(m' + r'p)\mathbf{1}_\star + [(m + rp)s' + (m' + r'p)s]\boldsymbol{a}_\star \\ &\quad + ss'\boldsymbol{a}_\star^{\circ 2} = (mm' + r_1 p)\mathbf{1}_\star + s_1 \boldsymbol{a}_\star + ss'\boldsymbol{a}_\star^{\circ 2} \mod pq, \end{aligned}$$

where $r_1 = mr' + m'r + rr'p$, $s_1 = (m + rp)s' + (m' + r'p)s$, and $\boldsymbol{a}_\star^{\circ 2} = [a_1^2 \ a_2^2 \ a_3^2]^T$.

As we can see, $\boldsymbol{z}_\star$ is not a valid encryption of $mm'$. We need to re-encrypt this product to eliminate the $\boldsymbol{a}_\star^{\circ 2}$ term.

We achieve this by multiplying $\boldsymbol{z}_\star$ by $R$. It is easy to check that $R\mathbf{1}_\star = \mathbf{1}$ and $R\boldsymbol{a}_\star = \boldsymbol{a}$, independently of $a_1, a_2$. Now

$$\begin{aligned}
(R\boldsymbol{a}_\star^{\circ 2})_1 &= (1-2\alpha_1)a_1^2 + \alpha_1 a_2^2 + \alpha_1(2a_1 - a_2)^2 \\
&= a_1^2 + \alpha_1((2a_1 - a_2)^2 + a_2^2 - 2a_1^2) \\
&= a_1^2 + 2\alpha_1(a_2 - a_1)^2 \\
&= a_1^2 + \alpha_1\beta \\
&= \varrho p + \sigma a_1 \\
(R\boldsymbol{a}_\star^{\circ 2})_2 &= -2\alpha_2 a_1^2 + (\alpha_2 + 1)a_2^2 + \alpha_2(2a_1 - a_2)^2 \\
&= a_2^2 + \alpha_2((2a_1 - a_2)^2 + a_2^2 - 2a_1^2) \\
&= a_2^2 + 2\alpha_2(a_2 - a_1)^2 \\
&= a_2^2 + \alpha_2\beta \\
&= \varrho p + \sigma a_2
\end{aligned}$$

Thus, we obtain the identity $R\boldsymbol{a}_\star^{\circ 2} = \varrho p\mathbf{1} + \sigma\boldsymbol{a}$.

So, applying $R$ to $\boldsymbol{z}_\star$, i.e. $\boldsymbol{z}' = R\boldsymbol{z}_\star$, gives

$$\begin{aligned}
\boldsymbol{z}' &= (mm' + r_1 p)R\mathbf{1} + s_1 R\boldsymbol{a} + ss' R\boldsymbol{a}^{\circ 2} \\
&= (mm' + r_1 p)\mathbf{1} + s_1\boldsymbol{a} + ss'(\sigma\boldsymbol{a} + \varrho p\mathbf{1}) \\
&= (mm' + r_2 p)\mathbf{1} + (s_1 + \sigma r r')\boldsymbol{a} \\
&= (mm' + r_2 p)\mathbf{1} + s_2\boldsymbol{a} \qquad (\mathrm{mod}\ pq)
\end{aligned}$$

for some integers $r_2, s_2$. So $\boldsymbol{z}'$ is a valid encryption of $mm'$. □

**Theorem 5.** *SPACDP is of equivalent complexity to the special case of HE2 where $\delta = a_2 - a_1$ ($0 < \delta < q$) is known.*

*Proof.* Suppose we have a system of $n$ approximate prime multiples, $m_i + r_i p$ ($i = 1, 2, \ldots, n$). Then we generate values $a, s_1, s_2, \ldots, s_n \xleftarrow{\$} [0, pq)$, and we have an oracle set up the cryptosystem with $a_1 = a$, $a_2 = a + \delta$. The oracle has access to $p$ and provides us with $R$, but no information about its choice of $\varrho$ and $\sigma$. We then generate the ciphertexts $\boldsymbol{c}_i$ ($i = 1, 2, \ldots, n$):

$$\begin{bmatrix} c_{i1} \\ c_{i2} \end{bmatrix} = \begin{bmatrix} m_i + r_i p + s_i a \\ m_i + r_i p + s_i(a + \delta) \end{bmatrix} \quad (\mathrm{mod}\ pq). \tag{9}$$

Thus $c_{i1} - s_i a = c_{i2} - s_i(a + \delta) = m_i + r_i p$. Thus finding the $m_i$ in (9) in polynomial time solves SPACDP in polynomial time.

Conversely, suppose we have any HE2 system with $a_2 = a_1 + \delta$. The ciphertext for $m_i$ ($i = 1, 2, \ldots, n$) is as in (9). so $s_i = \delta^{-1}(c_{i2} - c_{i1})$. Since

$0 < \delta < q < p$, $\delta$ is coprime to both $p$ and $q$, and hence $\delta^{-1} \mod pq$ exists. Thus breaking the system is equivalent to determining the $m_i \mod p$ from $m_i+\delta^{-1}(c_{i2}-c_{i1})a+r_ip$ $(i = 1, 2, \ldots, n)$. Determining the $m_i+\delta^{-1}(c_{i2}-c_{i1})a$ from the $m_i+\delta^{-1}(c_{i2}-c_{i1})a+r_ip$ $(i = 1, 2, \ldots, n)$ can be done using SPACDP. However, we still need to determine $a$ in order to determine $m_i$. This can be done by "deciphering" $R$ using SPACDP. We have

$$2\delta^2\alpha_1 = \sigma a - a^2 + \varrho p, \qquad 2\delta^2\alpha_2 = \sigma(a+\delta) - (a+\delta)^2 + \varrho p,$$

so $\sigma = 2\delta^2(\alpha_2 - \alpha_1) - 2ka - \delta^2$. Now $a$ can be determined by first determining $m_0 = a(2\delta^2(\alpha_2 - \alpha_1) - (2\delta + 1)a - \delta^2)$ from $m_0 + \varrho p = 2\delta^2\alpha_1$. This can be done using SPACDP. Then $a$ can be determined by solving the quadratic equation $m_0 = a(2\delta^2(\alpha_2 - \alpha_1) - (2\delta + 1)a - \delta^2) \mod p$ for $a$. This can be done probabilistically in polynomial time using, for example, the algorithm of Berlekamp [7]. So the case $\boldsymbol{a} = [a \;\; a+\delta]^T$, with known $\delta$, can be attacked using SPACDP on the system

$$m_0 + \varrho p,\; m_1 + \delta^{-1}(c_{11} - c_{12})a + r_1 p,$$
$$\ldots,\; m_n + \delta^{-1}(c_{n1} - c_{n2})a + r_n p. \qquad \square$$

**Lemma 2.** $\Pr(\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{k-1}$ *do not form a basis*$) \leq (k-1)(1/p + 1/q)$.

*Proof.* The $\boldsymbol{a}$'s are a basis if $A_k^{-1}$ exists, since then $\boldsymbol{v} = A_k\boldsymbol{r}$ when $\boldsymbol{r} = A_k^{-1}\boldsymbol{v}$, for any $\boldsymbol{v}$. Now $A_k^{-1}$ exists $\bmod pq$ if $(\det A_k)^{-1} \mod pq$ exists, by constructing the adjugate of $A_k$. Now $(\det A_k)^{-1} \mod pq$ exists if $\det A_k \neq 0 \mod p$ and $\det A_k \neq 0 \mod q$. Now $\det A_k$ is a polynomial of total degree $(k-1)$ in the $a_{ij}$ $(0 < i \leq k, 0 < j < k)$, and is not identically zero, since $\det A_k = 1$ if $\boldsymbol{a}_i = \boldsymbol{e}_{i+1}$ $(1 < i < k)$. Also $a_{ij} \xleftarrow{\$} [0, pq)$ implies $a_{ij} \bmod p \xleftarrow{\$} [0, p)$ and $a_{ij} \bmod q \xleftarrow{\$} [0, q)$. Hence, using the Schwartz-Zippel Lemma (SZL) [44], we have $\Pr(\det A_k = 0 \bmod p) \leq (k-1)/p$ and $\Pr(\det A_k = 0 \bmod q) \leq (k-1)/q$, and it follows that $\Pr(\nexists\, (\det A_k)^{-1} \bmod pq) \leq (k-1)(1/p + 1/q)$. $\square$

**Lemma 3.** *Let* $A_{\star k} = [\boldsymbol{a}_{\star 0}\; \boldsymbol{a}_{\star 1}\; \ldots\; \boldsymbol{a}_{\star,k-1}]$*, where the columns of* $A_k$ *form a basis for* $\mathbb{Z}_{pq}^k$*. If* $RA_{\star k} = A_k$*, then* $R\boldsymbol{v}_\star = \boldsymbol{v}$ *for all* $\boldsymbol{v} \in \mathbb{Z}_{pq}^k$*.*

*Proof.* We have $\boldsymbol{v} = A_k\boldsymbol{r}$ for some $\boldsymbol{r} \in \mathbb{Z}_{pq}^k$. Then $A_{\star k} = U_k A_k$ and $\boldsymbol{v}_{\star k} = U_k\boldsymbol{v}$, so $R\boldsymbol{v}_\star = RU_k\boldsymbol{v} = RU_kA_k\boldsymbol{r} = RA_{\star k}\boldsymbol{r} = A_k\boldsymbol{r} = \boldsymbol{v}$. $\square$

**Theorem 6.** $A_{\star k}^{\circ 2}$ *has no inverse* $\bmod pq$ *with probability at most* $(k^2-1)(1/p + 1/q)$.

*Proof.* We use the same approach as in Lemma 2. Thus $A_{\star k}^{\circ 2}$ is invertible provided $\det A_{\star k}^{\circ 2} \neq 0 \mod p$ and $\det A_{\star k}^{\circ 2} \neq 0 \mod q$. Let $\boldsymbol{A}$ denote the vector of

$a_{ij}$'s, $(a_{ij} : 1 \le i \le k, 1 \le j < k)$. The elements of $A^{\circ 2}_{\star k}$ are quadratic polynomials over $\boldsymbol{A}$, except for the first column, which has all 1's, and columns $2, 3, \ldots, k$ which are linear polynomials. So $\det A^{\circ 2}_{\star k}$ is a polynomial over $\boldsymbol{A}$ of total degree $2\binom{k}{2} + k - 1 = k^2 - 1$. Thus, unless $\det A^{\circ 2}_{\star k}$ is identically zero as a polynomial over $\boldsymbol{A}$, the SZL [44] implies $\Pr(\nexists\, (\det A^{\circ 2}_{\star k})^{-1} \bmod p) \le (k^2 - 1)/p$ and $\Pr(\nexists\, (\det A^{\circ 2}_{\star k})^{-1} \bmod q) \le (k^2 - 1)/q$. Therefore we have $\Pr(\nexists\, (\det A^{\circ 2}_{\star k})^{-1} \bmod pq) \le (k^2 - 1)(1/p + 1/q)$.

It remains to prove that $\det A^{\circ 2}_{\star k}$ is not identically zero as a polynomial over $\boldsymbol{A}$ in either $\mathbb{Z}_p$ or $\mathbb{Z}_q$. We prove this by induction on $k$. Consider $\mathbb{Z}_p$, the argument for $\mathbb{Z}_q$ being identical. Since $\mathbb{Z}_p$ is a field, $\det A^{\circ 2}_{\star k}$ is identically zero if and only if it has rank less than $\binom{k+1}{2}$ for all $\boldsymbol{A}$. That is, there exist $\lambda_{ij}(\boldsymbol{A}) \in \mathbb{Z}_p$ $(0 \le i \le j < k)$, not all zero, so that

$$\begin{aligned}\mathcal{L}(\boldsymbol{A}) &= \sum_{0 \le i \le j}^{k-1} \lambda_{ij} \boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j} \\ &= \boldsymbol{\alpha} + \boldsymbol{a}_{\star,k-1} \circ \boldsymbol{\beta} + \lambda_{k-1,k-1} \boldsymbol{a}^{\circ 2}_{\star,k-1} \;= 0,\end{aligned}$$

where $\boldsymbol{\alpha} = \sum_{0 \le i \le j}^{k-2} \lambda_{ij} \boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}$ and $\boldsymbol{\beta} = \sum_{i=0}^{k-2} \lambda_{i,k-1} \boldsymbol{a}_{\star i}$ are independent of $\boldsymbol{a}_{\star,k-1}$.

Clearly $\lambda_{k-1,k-1} = 0$. Otherwise, whatever $\boldsymbol{\alpha}, \boldsymbol{\beta}$, we can choose values for $\boldsymbol{a}_k$ so that $\mathcal{L} \ne 0$, a contradiction. Now suppose $\lambda_{i,k-1} \ne 0$ for some $0 \le i < k - 1$. The matrix $\hat{A}_\star$ with columns $\boldsymbol{a}_{\star i}$ $(0 \le i < k - 1)$ contains $A_{k-1}$ as a submatrix, which has rank $(k - 1)$ with high probability by Lemma 2. Thus $\beta \ne \boldsymbol{0}$ and, whatever $\boldsymbol{\alpha}$, we can choose values for $\boldsymbol{a}_k$ so that $\mathcal{L} \ne 0$. Thus $\lambda_{i,k-1} = 0$ for all $0 \le i < k$. Thus $\lambda_{ij} \ne 0$ for some $0 \le i \le j < k - 1$. Now the matrix $\hat{A}^{\circ 2}_\star$ with $\binom{k}{2}$ columns $\boldsymbol{a}_{\star i} \circ \boldsymbol{a}_{\star j}$ $(0 \le i \le j < k - 1)$ contains $A^{\circ 2}_{\star,k-1}$ as a submatrix, and therefore has rank $\binom{k}{2}$ by induction. Hence $\alpha \ne \boldsymbol{0}$, implying $\mathcal{L} \ne 0$, a contradiction. □

**Lemma 4.** $\Pr(\det C = 0 \bmod pq) \le (2k - 1)(1/p + 1/q)$.

*Proof.* From Lemma 2, $\det A = 0 \mod p$ or $\det A = 0 \mod q$ with probability at most $(k - 1)(1/p + 1/q)$. So $\det A$ is not zero or a divisor of zero $\bmod pq$. The entries of $W'$ are random $[0, pq)$, and $\det W'$ is a polynomial of total degree $k$ in its entries. It is a nonzero polynomial, since $W' = I_k$ is possible. Hence, using the SZL [44], $\Pr(\det W' = 0 \bmod p) \le k/p$ and $\Pr(\det W' = 0 \bmod q) \le k/q$. So $\det W'$ is zero or a divisor of zero $\bmod pq$ with probability at most $k(1/p + 1/q)$. So $\det A \det W' = 0 \bmod pq$ with probability at most $(2k - 1)(1/p + 1/q)$. So $\det C \ne 0$ with high probability. □

**Lemma 5.** $\Pr(\det C_0 = 0 \bmod pq) \le (2k - 1)(1/p + 1/q)$.

*Proof.* Note that $C_0 = C$ if $m_1 = m_2 = \cdots = m_k = 0$. Since Lemma 4 holds in that case, the result follows. □

**Lemma 6.** *Equation* (7) *holds if and only if* $\sum_{t=1}^{k-1} \sigma_{jlt}\varrho_{it} = \sum_{t=1}^{k-1} \sigma_{ijt}\varrho_{lt} \pmod q$, $\forall i, j, l \in [1, k-1]$.

*Proof.* Since $\boldsymbol{\gamma}^T\mathbf{1} = 1$ and $\boldsymbol{\gamma}^T\boldsymbol{a}_i = 0, i \in [1, k-1]$, $\boldsymbol{\gamma}^T(\boldsymbol{a}_i \cdot \boldsymbol{a}_j) = \boldsymbol{\gamma}^T\big(p\varrho_{ij}\mathbf{1} + \sum_{l=1}^{k-1} \sigma_{ijl}\boldsymbol{a}_l\big) = p\varrho_{ij}$. Thus

$$\begin{aligned}\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l) &= \boldsymbol{a}_i \cdot \big(p\varrho_{jl}\mathbf{1} + \sum_{t=1}^{k-1} \sigma_{jlt}\boldsymbol{a}_t\big) \\ &= p\varrho_{jl}\boldsymbol{a}_i + \sum_{t=1}^{k-1} \sigma_{jlt}\boldsymbol{a}_i \cdot \boldsymbol{a}_t,\end{aligned}$$

and hence $\boldsymbol{\gamma}^T[\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l)] = p \sum_{t=1}^{k-1} \sigma_{jlt}\varrho_{it}$. Similarly $\boldsymbol{\gamma}^T[(\boldsymbol{a}_i \cdot \boldsymbol{a}_j) \cdot \boldsymbol{a}_l] = p \sum_{t=1}^{k-1} \sigma_{ijt}\varrho_{lt}$, and the lemma follows. □

**Lemma 7.** *Let* $\tau, \varrho_i \xleftarrow{\$} [0, q)$ $(i \in [1, k-1])$*, let* $\varrho_{ij} = \varrho_i\varrho_j \mod q$*, and let the* $\sigma_{ijl}$ *satisfy* $\sum_{l=1}^{k-1} \sigma_{ijl}\varrho_l = \tau\varrho_i\varrho_j \pmod q$ *for all* $i, j \in [1, k-1]$*. Then, for all* $i, j, \ell \in [1, k-1]$, $\boldsymbol{\gamma}^T(\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l)) = \tau\varrho_i\varrho_j\varrho_l \mod q$*, the symmetry of which implies* (7)*.*

*Proof.* We have $\boldsymbol{\gamma}^T(\boldsymbol{a}_j \cdot \boldsymbol{a}_l) = p\varrho_{ij} = p\varrho_j\varrho_l$ for all $j, \ell \in [1, k-1]$. Hence, $\bmod q$,

$$\begin{aligned}\boldsymbol{\gamma}^T(\boldsymbol{a}_i \cdot (\boldsymbol{a}_j \cdot \boldsymbol{a}_l)) &= p \sum_{t=1}^{k-1} \sigma_{jlt}\varrho_{it} \\ &= p \sum_{t=1}^{k-1} \sigma_{jlt}\varrho_i\varrho_t \\ &= p\varrho_i \sum_{t=1}^{k-1} \sigma_{jlt}\varrho_t \\ &= p\varrho_i\tau\varrho_j\varrho_l = p\tau\varrho_i\varrho_j\varrho_l.\end{aligned}$$

□

## C Derivation of Bounds

To recap, $n$ is the number of inputs, $M$ is an exclusive upper bound on the inputs, $d$ is the degree of the polynomial we wish to calculate. We take $p \approx 2^\lambda$ and then $q \approx 2^\eta$, where $\eta = \lambda^2/\rho - \lambda$, to guard against the attacks of [17, 36].

For HE1, we assume $M \approx 2^\rho$, $n \leq \sqrt{M}$. Therefore,

$$p > (n+1)^d M^d \approx (nM)^d \text{ for large } n.$$

So, we may take

$$p = 2^\lambda > M^{3d/2} \approx 2^{3d\rho/2} \quad \text{i.e. } \lambda \approx 3d\rho/2$$
$$\text{and } \eta \approx \frac{\lambda^2}{\rho} - \lambda = \frac{3d\lambda}{2} - \lambda = \frac{3d\rho}{2}\left(\frac{3d}{2} - 1\right)$$

For HE1N, we assume $M \approx 2^\rho$, and we have $\rho' = \rho + \lg \kappa$. Now,

$$\kappa > (n+1)^d M^d \approx (nM)^d \text{ for large } n, \quad \text{i.e. } \lg \kappa \approx d(\lg n + \rho)$$

Therefore, since $\rho = \rho' - \lg \kappa$,

$$\lg \kappa > d \lg n + d(\rho' - \lg \kappa) \quad \text{i.e. } \lg \kappa \approx \frac{d(\lg n + \rho')}{d+1}$$

Since $\kappa$ is much larger than $M$, we also have

$$p = 2^\lambda > (n+1)^d (M + \kappa^2)^d \approx (n\kappa^2)^d \text{ for large } n \quad \text{i.e. } \lambda \approx d(\lg n + 2 \lg \kappa),$$
$$\text{and } \eta \approx \frac{\lambda^2}{\rho'} - \lambda = \frac{3d\lambda}{2} - \lambda = \frac{3d\rho'}{2}\left(\frac{3d}{2} - 1\right)$$

Then we can calculate $\eta$ as for HE1 above. Note that, in both HE1 and HE1N, $\lambda$ scales linearly with $d$, and $\eta$ scales quadratically. These bounds carry over to HE2, HE2N, HE$k$ and HE$k$N.

## References


[1] A. Acar et al. *A Survey on Homomorphic Encryption Schemes: Theory and Implementation*. 2017. arXiv: 1704.03578 [cs.CR].

[2] C. Aguilar-Melchor et al. 'A Comparison of Open-Source Homomorphic Libraries With Multi-Precision Plaintext Moduli'. WHEAT 2016. July 2016. URL: https://wheat2016.lip6.fr/ricosset.pdf.

[3] J.-P. Aumasson. *On the pseudo-random generator ISAAC*. 2006. Cryptology ePrint Archive: 2006/438.

[4] M. Bellare and P. Rogaway. 'Introduction to Modern Cryptography'. Lecture Notes. 2005.

[5] M. Bellare et al. 'A Concrete Security Treatment of Symmetric Encryption'. In: *Proc. FOCS '97*. 1997, pp. 394–403.

[6] M. Bellare et al. 'Relations Among Notions of Security for Public-Key Encryption Schemes'. In: *Proc. CRYPTO '98*. 1998, pp. 26–45.

[7] E. R. Berlekamp. 'Factoring Polynomials Over Large Finite Fields'. In: *Mathematics of Computation* 24.111 (1970), pp. 713–735.

[8] S. Bogos et al. *Cryptanalysis of a Homomorphic Encryption Scheme*. 2016. Cryptology ePrint Archive: 2016/775.

[9] D. Boneh and V. Shoup. 'A Graduate Course in Applied Cryptography'. Draft 0.2. 2015.

[10] C. Bonte et al. *Faster Homomorphic Function Evaluation using Non-Integral Base Encoding*. 2017. Cryptology ePrint Archive: 2017/333.

[11] J. W. Bos et al. *Privacy-friendly Forecasting for the Smart Grid using Homomorphic Encryption and the Group Method of Data Handling*. 2016. Cryptology ePrint Archive: 2016/1117.

[12] Z. Brakerski and V. Vaikuntanathan. 'Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages'. In: *Proc. CRYPTO 2011*. 2011, pp. 505–524.

[13] Z. Brakerski et al. '(Leveled) Fully Homomorphic Encryption Without Bootstrapping'. In: *Proc. ITCS '12*. 2012, pp. 309–325.

[14] D. Catalano and D. Fiore. *Boosting Linearly-Homomorphic Encryption to Evaluate Degree-2 Functions on Encrypted Data*. 2014. Cryptology ePrint Archive: 2014/813.

[15] D. Catalano and D. Fiore. 'Using Linearly-Homomorphic Encryption to Evaluate Degree-2 Functions on Encrypted Data'. In: *Proc. CCS '15*. ACM, 2015, pp. 1518–1529.

[16] Y. Chen and P. Q. Nguyen. 'Faster Algorithms for Approximate Common Divisors: Breaking Fully Homomorphic Encryption Challenges over the Integers'. In: *Proc. EUROCRYPT '12*. 2012, pp. 502–519.

[17] H. Cohn and N. Heninger. 'Approximate common divisors via lattices'. In: *Proc. ANTS-X*. Vol. 1. 2012, pp. 271–293.

[18] D. Coppersmith. 'Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities'. In: *J. Cryptology* 10.4 (1997), pp. 233–260.

[19] J.-S. Coron et al. 'Fully Homomorphic Encryption over the Integers with Shorter Public Keys'. In: *Proc. CRYPTO '11*. 2011, pp. 487–504.

[20] CryptoExperts. *FV-NFLib*. URL: https://github.com/CryptoExperts/FV-NFLlib.

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

[22] M. van Dijk et al. 'Fully Homomorphic Encryption over the Integers'. In: *Proc. EUROCRYPT '10*. 2010, pp. 24–43.

[23] M. van Dijk and A. Juels. 'On the Impossibility of Cryptography Alone for Privacy-Preserving Cloud Computing'. In: *Proc. HotSec '10*. 2010, pp. 1–8.

[24] L. Ducas and D. Micciancio. 'FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second'. In: *Proc. EUROCRYPT '15*. 2015, pp. 617–640.

[25] L. Ducas and D. Micciancio. *FHEW. A Fully Homomorphic Encryption Library*. URL: https://github.com/lducas/FHEW.

[26] T. ElGamal. 'A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms'. In: *IEEE Trans. Inf. Theory* 31.4 (1985), pp. 469–472.

[27] Z. Erkin et al. 'Privacy-Preserving Face Recognition'. In: *Proc. PETS '09*. 2009, pp. 235–253.

[28] J. Fan and F. Vercauteren. *Somewhat Practical Fully Homomorphic Encryption*. 2012. Cryptology ePrint Archive: 2012/144.

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

[30] C. Gentry. 'Fully Homomorphic Encryption Using Ideal Lattices'. In: *Proc. STOC '09*. 2009, pp. 169–178.

[31] O. Goldreich et al. 'How to Play ANY Mental Game'. In: *Proc. STOC '87*. 1987, pp. 218–229.

[32] S. Goldwasser and S. Micali. 'Probabilistic Encryption'. In: *J. Comput. Syst. Sci.* 28.2 (1984), pp. 270 –299.

[33] S. Goldwasser et al. 'Reusable Garbled Circuits and Succinct Functional Encryption'. In: *Proc. STOC '13*. 2013, pp. 555–564.

[34] S. Halevi and V. Shoup. 'Bootstrapping for HElib'. In: *Proc. EUROCRYPT '15*. 2015, pp. 641–670.

[35] S. Halevi and V. Shoup. *HELib*. URL: https://github.com/shaih/HElib.

[36] N. Howgrave-Graham. 'Approximate Integer Common Divisors'. In: *Cryptography and Lattices*. 2001, pp. 51–66.

[37] B. Jenkins. *ISAAC: a fast cryptographic random number generator*. 1996. URL: http://burtleburtle.net/bob/rand/isaacafa.html.

[38] M. Joye and B. Libert. 'Efficient Cryptosystems from $2^k$-th Power Residue Symbols'. In: *Proc. EUROCRYPT '13*. 2013, pp. 76–92.
[39] A. Kipnis and E. Hibshoosh. *Efficient Methods for Practical Fully Homomorphic Symmetric-key Encryption, Randomization and Verification*. 2012. Cryptology ePrint Archive: 2012/637.
[40] T. Kleinjung et al. 'Factorization of a 768-bit RSA Modulus'. In: *Proc. CRYPTO '10*. 2010, pp. 333–350.
[41] K. Laine et al. *Simple Encrypted Arithmetic Library - SEAL*. Version 2.2. 2017. URL: https://sealcrypto.codeplex.com/.
[42] K. Lauter et al. 'Can Homomorphic Encryption Be Practical?' In: *Proc. CCSW '11*. 2011, pp. 113–124.
[43] J. L. Massey. 'Guessing and Entropy'. In: *Proc. ISIT '94*. 1994, p. 204.
[44] D. Moshkovitz. 'An Alternative Proof of the Schwartz-Zippel Lemma.' In: *Electronic Colloquium on Computational Complexity (ECCC)*. 2010, p. 96.
[45] P. Paillier. 'Public-Key Cryptosystems Based on Composite Degree Residuosity Classes'. In: *Proc. EUROCRYPT '99*. 1999, pp. 223–238.
[46] R. A. Popa et al. 'CryptDB: Protecting Confidentiality With Encrypted Query Processing'. In: *Proc. SOSP '11*. 2011, pp. 85–100.
[47] M. O. Rabin. *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*. Tech. rep. MIT/LCS/TR-212. 1979, p. 12.
[48] T. Ricosset. *HElib-MP*. URL: https://github.com/tricosset/HElib-MP.
[49] R. L. Rivest et al. 'A Method for Obtaining Digital Signatures and Public-key Cryptosystems'. In: *Commun. ACM* 21.2 (1978), pp. 120–126.
[50] R. L. Rivest et al. 'On Data Banks and Privacy Homomorphisms'. In: *Foundations of Secure Computation* 4.11 (1978), pp. 169–180.
[51] R. D. Schafer. *An Introduction to Nonassociative Algebras*. Vol. 22. Dover, 1966.
[52] J. J. Stephen et al. 'Practical Confidentiality Preserving Big Data Analysis'. In: *Proc. HotCloud '14*. 2014, p. 10.
[53] S. D. Tetali et al. 'MRCrypt: Static Analysis for Secure Cloud Computations'. In: *Proc. OOPSLA '13*. 2013, pp. 271–286.
[54] I. Thomson. *Microsoft researchers smash homomorphic encryption speed barrier*. 9th Feb. 2016. URL: https://www.theregister.co.uk/2016/02/09/researchers_break_homomorphic_encryption/.
[55] M. Varia et al. *HETest: A Homomorphic Encryption Testing Framework*. 2015. Cryptology ePrint Archive: 2015/416.
[56] S. Vivek. *Homomorphic Encryption API Software Library*. 21st Feb. 2017. URL: http://heat-h2020-project.blogspot.co.uk/2017/02/homomorphic-encryption-api-software.html.
[57] D. Vizár and S. Vaudenay. 'Cryptanalysis of Chosen Symmetric Homomorphic Schemes'. In: *Stud. Sci. Math. Hung.* 52.2 (2015), pp. 288–306.
[58] A. Yu et al. *Efficient Integer Vector Homomorphic Encryption*. 2015. URL: https://courses.csail.mit.edu/6.857/2015/files/yu-lai-payor.pdf.
[59] H. Zhou and G. Wornell. 'Efficient homomorphic encryption on integer vectors and its applications'. In: *Proc. ITA '14*. 2014, pp. 1–9.