A Practical Implementation of Identity-Based Encryption over NTRU Lattices

Sarah McCarthy, Neil Smyth, and Elizabeth O'Sullivan

Centre for Secure Information Technologies (CSIT), Queen's University Belfast, UK
{smccarthy10, n.smyth, e.osullivan}@qub.ac.uk

Abstract. An identity-based encryption scheme enables the efficient distribution of keys in a multi-user system. Such schemes are particularly attractive in resource constrained environments where critical resources such as processing power, memory and bandwidth are severely limited. This research examines the first pragmatic lattice-based IBE scheme presented by Ducas, Lyubashevsky and Prest in 2014 and brings it into the realm of practicality for use on small devices. This is the first standalone ANSI C implementation of all the software elements of the scheme with improved performance. User Key Extraction demonstrates a 180% speed increase and Encrypt and Decrypt demonstrate increases of over 500% and 1200% respectively for 80-bit security on an Intel Core i7-6700 CPU at 4.0 GHz, with similar accelerations for 192-bit security, compared with Prest's NTL proof-of-concept implementation on an Intel Core i5-3210M CPU at 2.5GHz. In addition, we provide a range of suggestions to further enhance performance.

Keywords: lattice-based cryptography, identity-based encryption, NTRU

1 Introduction

Managing keys in systems that use Public Key Cryptography (PKC) as the primary means of authentication is achieved within the established framework of a Public Key Infrastructure (PKI). However, it is widely known that PKI requires an extensive network for operational and certificate management. This imposes significant overheads in resource constrained environments and in practice limits the frequency that PKC can be used for large scale, bandwidth intensive applications requiring real-time performance. As technology progresses to operational landscapes such as the Internet of Things (IoT), the need for alternative PKC schemes comes to the fore. Schemes such as Identity-based Encryption (IBE) have been introduced to overcome the complexity issues associated with traditional PKI based approaches. IBE simplifies key generation and distribution in a multi-user system. In environments with limited resources IBE can offer the potential for PKC to be utilised when it is needed and not just when it can be accommodated. In addition to resource efficient schemes, many applications have long-term security requirements. For such applications the threat of quantum attacks must be mitigated [NIST-IR8109]. The formulation of Shor's algorithm over two decades ago has prompted research into mathematical areas which could potentially provide quantum-security. One such area is that of lattices and their associated NP-hard problems. Finding lattice-based schemes analogous to classical schemes such as public key encryption and digital signature schemes is the subject of intense research. One of the main advantages of lattice-based cryptography is that it can also provide quantum-resilient alternatives to today's IBE schemes, traditionally based on pairing. Ducas, Lyubashevsky and Prest proposed such a scheme in 2014 [11], henceforth referred to as DLP-IBE. This research presents a complete standalone ANSI C implementation of this scheme. There has been no prior indication of the practical performance and costs of a lattice-based IBE scheme. This work not only provides a benchmark for this type of scheme, but our efficiency improvements bring it into the realm of practicality for the post-quantum setting.

The paper is organised as follows: Section 1 reviews both non-lattice and lattice-based IBE schemes and gives a background of lattice geometry. Section 2 introduces the DLP-IBE scheme, and Section 3 describes the proposed software architecture, which includes the functionality of Master Key Generation and Extract algorithms as well as the Encrypt and Decrypt processes. Section 4 gives the conclusions. We acknowledge the major bottlenecks, optimisations and challenges and our methods to address them. Implementation results are also given along with potential extensions of the scheme.

1.1 Identity-Based Encryption

An IBE scheme is one where the user's public key is a piece of meaningful information, such as an email address, or a device identifier. A trusted authority uses a master secret key to compute the user secret key. The authority has a master public key which is also needed to send messages to the user. The use of already established personal information as the user ID removes the need for public key distribution. Elements such as timestamps can be incorporated into the user keys to provide a key refresh mechanism. Additionally, the use of timestamps with the user ID can allow senders to encrypt messages that can only be read in the future. In the generic IBE instantiation the central authority has complete access to keys and can therefore decrypt any message and so should be trusted. Additionally, the communication channels between users and the trusted authority should be secure.

The first notion of identity-based schemes was presented by Shamir in 1984 [26]. The idea was to eliminate the need for a public certificate across email systems. These schemes allowed secure communication without exchanging user keys. Shamir presented a solution for an identity-based signature scheme but it wasn't until 2001 that such an encryption scheme was realised [5]. There are two main threads of constructing traditional IBE schemes; using pairings or quadratic residues. The most prominent schemes are Boneh-Franklin [5], Cocks [10] and Sakai-Kasahara [25]. Boneh co-founded a start-up company in 2002 called Voltage Security Inc. [1] which currently provides IBE solutions to industry. These include secure email and file transfer applications. However, these schemes are susceptible to quantum attacks due to Shor's algorithm, creating the need for quantum-resilient variants.

The first application of lattices (Section 1.2) to IBE schemes was in 2008 by Gentry et al. [14]. The main contribution of this work was a sampling algorithm (known as GPV sampling) which showed how to use a short basis as a trapdoor for generating short lattice vectors. This sampler was then used to construct a lattice-based IBE scheme that resembled Cocks' traditional scheme (due to the use of a trapdoor), and can be considered as the dual of Regev's LWE scheme [23]. However, the security proof was in the random oracle model and the master public key and user secret keys had large sizes of $O(n^2)$ bits. In 2010, Agrawal et al. [1] proposed a Learning With Errors (LWE)-based IBE scheme with a trapdoor structure, with performance comparable to the GPV scheme. It uses a sampling algorithm to obtain a basis with low Gram-Schmidt Norm for the master secret key and forms a lattice family with two associated trapdoors to generate short vectors; one for all lattices in the family and the other for all but one. It improves on previous schemes which process the user identities bit by bit by instead considering them as a whole. The public key is $O(nm)$, where the lattice basis is of size n x m. In 2016, Apon et al. [3] proposed the most efficient *standard* LWE scheme to date with a public key size of $O(2nm\log(q))$. This includes the design of new encoding scheme for identities, incorporating a collision-resistant hash function. The first Ring-LWE based IBE scheme was the DLP-IBE scheme [11]. The use of the ring variant increases efficiency by reducing the public key to a polynomial/vector of $O(n)$ and ciphertext $O(2n)$. However, it makes additional assumptions to standard LWE. In particular, it uses the GPV sampling algorithm on a certain distribution of NTRU lattices to increase its efficiency. Other Ring-LWE schemes have since been proposed, for example in 2016 Katsumata and Yamada [16] introduced a scheme based on Yamada's 2016 standard-LWE scheme [29], and exploits the ring properties and assumes the Ring-LWE problem hardness for fixed polynomial approximation factors. The public parameters in this scheme are of size $O(nl^{1/d}\log(n))$, ciphertext $O(n\log(n))$ and private key $O(n\log(n))$. However, the DLP-IBE scheme is still considered the most efficient scheme to date due to smaller key sizes.

1.2 Lattice-based Cryptography

Shor's algorithm [27] has prompted the research community to investigate so-called "quantum-resistant" forms of cryptography. One of the strong contenders is lattice-based cryptography. The advantages of this type of cryptography are the associated "worst-case hardness" properties [2], efficiency of implementations and flexibility, as it has potential to be used in both encryption and digital signature schemes, as well as IBE, attribute-based encryption (ABE) and even

[1] https://www.voltage.com

fully homomorphic encryption (FHE), although in general the latter two schemes have not yet demonstrated practicality.

A lattice is a mathematical structure, defined by a collection of vectors called a basis, denoted $\mathcal{B}$. The points v of the lattice are all the possible linear combinations of the basis vectors with integer coefficients:

$$\mathcal{L} = \{v = a_1b_1 + a_2b_2 + ... + a_nb_n : a_i \in \mathbb{Z}, b_i \in \mathcal{B}\}$$

Informally, this can be thought of as an infinite arrangement of regularly spaced points. The closest vector problem (CVP), which is that of finding the closest lattice point to a given point in the space, is an NP-hard lattice problem. The shortest vector problem (SVP), which is that of finding the shortest vector in a lattice, is also NP-hard under randomised reductions. These and connected problems can be used as the basis of security for cryptographic schemes.

A popular lattice problem is the learning with errors (LWE) problem, formulated by Regev in 2005 [23] , and its ideal-lattice-based variant Ring-LWE [18]. Many cryptographic schemes based on these have been proposed, such as the encryption scheme in the original papers, digital signature scheme [19] and other concepts such as e-voting [9]. Adding structure and pattern to the lattice basis aids the working of the scheme. This can also improve memory and efficiency, for example by reducing the amount of basis information that needs to be stored or transported to recover the lattice. The DLP-IBE Scheme uses NTRU lattices for this reason. NTRU lattice bases have a convolutional, modular structure. The trapdoors in this scheme are the polynomials f, g, which allow the user to generate a "nice" basis $\mathcal{B}_{\text{nice}}$ whilst the public only have access to the lattice through a "bad" basis $\mathcal{B}_{\text{bad}}$ defined by polynomial h.

$$\mathcal{B}_{\text{bad}} = \begin{pmatrix} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{N-1} \\ 0 & 1 & \dots & 0 & -h_{n-1} & h_0 & \dots & h_{N-2} \\ \vdots & \ddots & & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 1 & -h_1 & -h_2 & \dots & h_0 \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \ddots & & \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{pmatrix} = \begin{pmatrix} 1 & \mathcal{H} \\ 0 & q \end{pmatrix} \text{ and } \mathcal{B}_{\text{nice}} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix},$$

where each $\mathcal{A}(\cdot)$ is an anti-circulant matrix dependent on a polynomial f, g, F or G and is of the form:

$$\mathcal{A}(f) = \begin{pmatrix} f_0 & f_1 & \dots & f_{N-1} \\ -f_{n-1} & f_0 & \dots & f_{N-2} \\ \vdots & & \ddots & \vdots \\ -f_1 & -f_2 & \dots & f_0 \end{pmatrix}$$

The NTRU lattice assumption is that it is a hard problem to recover polynomials f, g from h, where $h = g/f$, i.e. it is hard to obtain $\mathcal{B}_{\text{nice}}$ from $\mathcal{B}_{\text{bad}}$. The original NTRU system used the polynomial ring $\mathbb{Z}_q[x]/(x^N - 1)$, however the DLP-IBE scheme uses the NTRU distribution over the polynomial ring $\mathbb{Z}_q[x]/(x^N + 1)$, as proposed as by Stehlé and Steinfield in 2011 [28].

2 IBE Scheme Setup

The DLP-IBE scheme was introduced in [11] as the first efficient lattice-based IBE scheme, whereby the underlying computational hardness is the NTRU and the Ring-LWE assumption. The first practical efforts towards executing the DLP-IBE scheme was a Proof-of-Concept (PoC) implementation by the original authors [2]. This implementation is written in C++ and depends on the NTL library; it also relaxes some constraints such as security thresholds for the lattice basis size. Recently, Güneysu and Oder [15] demonstrated the efficiency of the Encrypt and Decrypt components of this scheme on a range of low-cost microcontrollers and reconfigurable hardware. The research in this paper implements and examines the entire scheme in C. In particular our focus is on the computationally intensive tasks of the Key Generation and Extraction algorithms. The design includes the enforcement of the aforementioned security thresholds and, in particular, provides the first indication of practical performance of the overall scheme.

IBE schemes consist of 4 main algorithms:
Master KeyGen: generates the master secret key and the master public key. Here, the private key is an NTRU lattice basis and the public key is an identifier of that lattice, in the form of a polynomial. See Section 3.1.
Extract: an algorithm to generate the user secret key, given their identity. It uses the master secret key and a specified hash function to do this. See Section 3.2.
Encryption and Decryption: encryption is the process public clients use to encrypt a message to a user. The DLP-IBE uses the generic ring-LWE encryption scheme [18]. The encryption process uniformly samples small error polynomials to encapsulate a uniformly-sampled key, the hash of which is used to one-time-pad the message. To decrypt, the key is recovered by rounding, and this allows the message to be output. See Section 3.3.

Table 1 shows the inputs and outputs of each algorithm in the DLP-IBE scheme. In DLP-IBE there is an additional Gaussian sampler algorithm (See Section 7).

Algorithm	**Inputs**	**Outputs**
Master KeyGen	N, q	$\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$, $h \in \mathcal{R}_q$
Extract	$\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$, $H : \{0,1\}^* \to \mathbb{Z}_q^N$, id	$\mathbf{SK}_{\mathrm{id}} \in \mathcal{R}_q$
Encryption	$h \in \mathcal{R}_q$, id, $m \in \{0,1\}^m$, $H : \{0,1\}^* \to \mathbb{Z}_q^N$, $H' : \{0,1\}^N \to \{0,1\}^m$.	$(u, v, c) \in \mathcal{R}_q^2$
Decryption	SK_{id}, $(u, v, c) \in \mathcal{R}_q^2$	$m \in \{0,1\}^N$

Table 1. Algorithm summary

[2] https://github.com/tprest/Lattice-IBE

2.1 Notation

Throughout the paper, we are working over the polynomial ring modulo (x^N+1) of integers modulo q, denoted $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^N+1)$. Here, N is a power of 2 and q is a prime congruent to $1 \mod 2N$. Vectors/polynomials are consider analogous and will be written as v. Matrices will denoted as $\mathbf{M}$. The lattice basis will be denoted by $\mathcal{B}$ or simply $\mathbf{B}$ depending on the context; the Gram-Schmidt Orthogonalisation denoted $\mathbf{B}_{gs}$ with basis vectors $\tilde{b_i}$. The Gaussian distribution over $\mathcal{R}_q$ with standard deviation σ is denoted $\mathcal{D}_{N,\sigma}$.

3 Software Design of the DLP-IBE Scheme

The DLP-IBE scheme is implemented in portable ANSI C. It is intended for general-purpose applications ranging from high-end 64-bit Intel Xeon servers to 32-bit ARM Cortex-M embedded systems. The design presented here focusses on 32-bit and 64-bit x86 processors in an Ubuntu/CentOS Linux environment, with options to configure client compilation on ARM v7 Cortex-A and Cortex-M target platforms. The Autotools build system is used to deliver the scheme as a library within a software distribution that can be suitably adapted to the host system at compile-time, i.e. utilising alternative algorithms for environments with constrained RAM. Additional adaptations can be configured at run-time, such as the selected underlying cryptographic functions (i.e. CSPRNG, hash), NTT optimisations, modular reduction techniques etc.

The proposed architecture considers a range of security levels to suit deployment needs, as in practice it is only possible to deploy particular levels of security on constrained devices, depending upon their capabilities, memory resources and the information being protected. For example, a battery-powered temperature sensor offering a 192-bit security strength would certainly be secure, but the reduced battery life and increased price of more capable hardware that is required are unlikely to appeal to consumers. More powerful devices within an typical IoT architecture may be able to support the full functionality of DLP-IBE, but peripheral devices may only support the much more efficient Encrypt and/or Decrypt functionality. In a standard scenario the peripheral devices could be provided with a user secret key for decryption purposes by one of three means: (a) embedded in firmware, (b) provided during device installation or (c) issued periodically by the DLP-IBE trusted authority. In such a scenario it is possible for DLP-IBE to provide public-key encryption for constrained devices.

The results have been obtained using GNU GCC 5.4.0. An Intel Core i7 6700 with both hyper-threading and TurboBoost disabled has been used, wherein the four CPUs are placed in performance mode at 4 GHz. GNU GMP 6.1.2 has been used to provide multiple precision arithmetic.

3.1 The KeyGen algorithm

The master key generation (KeyGen) algorithm generates the master keys. This happens once per environment setup. In this scheme, the KeyGen algorithm

(Algorithm 1) requires the degree N of defining polynomials f, g, h and the modulus q and outputs the master secret key $\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$ and master public key $h \in \mathcal{R}_q$.

Algorithm 1: Key Generation [11]

```
Data: $N, q$
Result: $\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$, $h \in \mathcal{R}_q$
1  $\sigma_f = 1.17\sqrt{\frac{q}{2N}}$
2  $f, g, \leftarrow \mathcal{D}_{N,\sigma_f}$
3  Norm $\leftarrow max\Big(||g, -f||, \Big|\Big|\Big(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}}\Big)\Big|\Big|\Big)$
4  if $Norm > 1.17\sqrt{q}$ then go to Step 2;
5  Compute $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ such that: $-\rho_f \cdot f = R_f$ and $-\rho_g \cdot g = R_g$
6  if $GCD(R_f, R_g) \neq 1$ or $GCD(R_f, q) \neq 1$ then go to Step 2;
7  Compute $u, v \in \mathbb{Z}$ such that: $u \cdot R_f + v \cdot R_g = 1$
8  $F \leftarrow qv\rho_g$ and $Q \leftarrow -qu\rho_f$
9  $k = \Big\lfloor \frac{F*\bar{f}+G*\bar{g}}{f*\bar{f}+g*\bar{g}} \Big\rceil \in \mathcal{R}$
10 $F \leftarrow F - k*f$ and $G \leftarrow G - k*g$
11 return $h = g * f^{-1} \mod q$
12 and $\mathbf{B} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$
```

In Step 1, the standard deviation of the Gaussian distribution from which f, g are generated is set to $\sigma_f = 1.17\sqrt{\frac{q}{2N}}$ and is chosen so that $E[||b_1||] = 1.17\sqrt{q}$. We have used a **CDT Sampler** in Step 2 to generate polynomials f, g from a discrete Gaussian distribution $\mathcal{D}_{N,\sigma_f}$ over $\mathcal{R}_q$. In Step 3, $||\tilde{B}_{f,g}||$, the **Gram-Schmidt Norm** of $B_{f,g}$, is computed, where $B_{f,g}$ is a basis of the NTRU lattice associated to f, g ($h = g * f^{-1} \mod q$). If $||\tilde{B}_{f,g}|| > 1.17\sqrt{q}$, the algorithm returns to Step 2 as the Gram-Schmidt Norm needs to be small enough so the basis can form a short trapdoor for sampling elements. In Step 5 the **Extended Euclidean Algorithm** is used to compute $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ such that $\rho_f \cdot f = R_f$ and $\rho_g \cdot g = R_g$. If $GCD(R_f, R_g) \neq 1$ or $GCD(R_f, q) \neq 1$, the algorithm returns to Step 2. Next, the algorithm computes $u, v \in \mathbb{Z}$ such that: $u \cdot R_f + v \cdot R_g = 1$. These integers are obtained from the Extended Euclidean algorithm (this extended version keeps track of the coefficients). In step 8, $F = qv\rho_g$ and $Q = -qu\rho_f$ is computed so that $f*G - g*F = q$, a condition needed to find a short basis. Next $k = \Big\lfloor \frac{F*\bar{f}+G*\bar{g}}{f*\bar{f}+g*\bar{g}} \Big\rceil$ is computed and F and G are reduced: $F = F - k*f$ and $G = G - k*g$. The final steps generate and output the keys. Polynomial $h = g * f^{-1} \mod q$ is the master public key and defines a lattice $\Lambda_{h,q}$. Matrix $\mathbf{B} = \begin{pmatrix} \mathcal{A}(g) & -\mathcal{A}(f) \\ \mathcal{A}(G) & -\mathcal{A}(F) \end{pmatrix}$ is the master secret key and is a short basis for $\Lambda_{h,q}$. $\mathcal{A}$ is an anti-circulant matrix defined previously in Section 1.2.

Key Generation is the most intensive component of the DLP-IBE scheme due to the arithmetic involving multiple-precision polynomials. The main software operations are generating the basis and multiple-precision arithmetic, using NTT and entropy coding. Optimal performance (i.e. the first f and g polynomials that are randomly selected are within bounds) is now ≈0.3 seconds slower than Prest's PoC implementation (i.e. 2.7 seconds vs 2.4 seconds). However, as the PoC was for reference purposes, it does not fully implement the f and g selection, which means that it generates keys that do not meet the security criteria and will not perform many retries for key selection if the bound thresholds are not met. The design proposed here enforces this security threshold.

Throughout the entire implementation, floating-point Barrett reduction is used with a precomputed inverse of q, i.e. multiply by the inverse of q, truncate towards zero and then multiply by q, subtracting the result from the input to obtain the remainder.

Gram-Schmidt Norm A lattice basis can have the Gram-Schmidt process applied to it. This reduces and shifts the basis vectors in relation to each other so they become shorter and more orthogonal, yet still define the same vector space ($\mathcal{B} \rightarrow \tilde{\mathcal{B}}$ and $Span(\mathcal{B}) = Span(\tilde{\mathcal{B}})$). The Gram-Schmidt Norm is a property of the basis. It is the maximum of the norms (moduli) of the vectors in the Gram-Schmidt orthogonalisation of the basis.

$$\text{GS Norm of } \mathcal{B} = ||\tilde{\mathcal{B}}|| = \max_{i \in I} ||\tilde{b_i}||$$

The obvious way to compute the Gram-Schmidt Norm would be to compute the norms of each of the vectors and take the maximum. However, in the case of the NTRU lattices, it was proved in Section 3.2 of [11] that there are only two candidate vectors with the largest norm, namely b_1 and $\tilde{b}_{N+1}$ (with the vectors ordered as in the definition of $\mathbf{B}$). Further to this, we can prove that $||\tilde{b}_{N+1}|| = \left|\left|\left(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}}\right)\right|\right|$. It is always the case that $\tilde{b}_1 = b_1$, therefore $||\tilde{b}_1|| = ||b_1||$. In this NTRU lattice basis, $b_1 =$ the top row of $(\mathcal{A}(g), -\mathcal{A}(f)) = (g, -f)$, therefore we can compute the Gram-Schmidt norm solely from f and g and form an alternative definition.

$$\text{GS Norm of } \mathcal{B} = \max_{i \in I} \left\{ ||b_1||, \left|\left|\left(\frac{q\bar{f}}{f*\bar{f}+g*\bar{g}}, \frac{q\bar{g}}{f*\bar{f}+g*\bar{g}}\right)\right|\right| \right\}$$

The Gram-Schmidt Norm computation is the first main bottleneck in the Key Generation algorithm. The b_1 norm calculation is simply placing two vectors in tandem and computing the dot product. However, the $\tilde{b}_{N+1}$ norm computation is more intensive as it involves polynomial multiplication over a ring, with polynomials coefficients being of approximately 2000-4000 bits in length. Therefore, we use the Number Theoretic Transform (NTT) to transform the polynomials into the polynomial ring of integers modulo p, meaning multiplication can be done coefficient-wise by reducing it to a negative wrapped convolution, rather than the more complex classical school-book method.

The Extended Euclidean Algorithm The Extended Euclidean Algorithm computes the Greatest Common Divisor (GCD) of two numbers (or polynomials) x, y, and the corresponding Bezout coefficients u, v, such that $ux + vy = GCD(x, y)$. Note the "extended" version refers to the algorithm in which the coefficients are both computed and stored. The KeyGen algorithm uses two versions: one for integer inputs and one for polynomial inputs. The Extended Euclidean Algorithm for polynomials is along the same principles as the regular version for integers. The differences are that the variables are polynomials, the input polynomials are divided by their leading coefficients in order to become monic for use in the algorithm, and (the more intensive) polynomial multiplication and division is used.

Step 5 of the Algorithm 1 states that given $f, g \in \mathcal{R}$, find $\rho_f, \rho_g \in \mathcal{R}$ and $R_f, R_g \in \mathbb{Z}$ which satisfy Equations 1 and 2.

$$-\rho_f \cdot f = R_f \mod (x^N + 1) \tag{1}$$

$$-\rho_g \cdot g = R_g \mod (x^N + 1) \tag{2}$$

The Extended Euclidean Algorithm for polynomials is used twice here. The first time, it is used to find the GCD of f and $x^N + 1$. The second time, it is used to find the GCD of g and $x^N + 1$. During the computation, the algorithm holds the coefficients while calculating the GCD of two integers or polynomials. The PoC reference implementation uses the XGCD function from NTL, which selects the most suitable strategy considering polynomial properties such as coefficient bit-length and degree. Here, we have used Brown's Modular GCD for computing GCDs of multi-precision polynomials, which consists of two subroutines. First, it maps the polynomials into the bivariate polynomial ring of integers modulo q, $\mathbb{Z}_q[x, y]$, and then uses the Chinese Remainder Theorem (CRT) to compute the coefficients.

A further subroutine is required to map these polynomials from $\mathbb{Z}_q[x, y] \approx \mathbb{Z}_q[x][y]$ into $\mathbb{Z}_q[x]$, perform the GCD computation and recover the y terms using CRT. The GCD computations within the subroutines are computed using Euclid's algorithm or a variant called Half-GCD, which recursively runs halfway through the Euclidean algorithm and uses the intermediate polynomials to reduce the original ones. Equation 3 illustrates the mathematical problem in the familiar format of $ux + vy = GCD(x, y)$, where u and v are the Bezout coefficients.

$$-\rho_f \cdot f + \triangle \cdot (x^N + 1) = R_f \tag{3}$$

The $\triangle$ represents one of the Bezout coefficients computed during the Extended Euclidean Algorithm but it becomes obsolete as we apply $\mod (x^N + 1)$ to each side to obtain Equation 4.

$$-\rho_f \cdot f = R_f \mod (x^N + 1) \quad (4)$$

The useful outputs here are the other Bezout coefficient ρ_f and the greatest common divisor R_f. The second use of the algorithm is similar but for g instead of f to obtain ρ_g and R_g. Currently GMP is used to provide the multiple-precision arithmetic, but this has been segmented in the software to a collection of wrapper functions to allow it to be replaced for future optimisations within the Extended GCD, multiplication and division components.

3.2 Extract

The Extract algorithm generates the user secret key for a given user ID. This algorithm is run once per user. In this scheme, the Extract Algorithm requires the master secret key $\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$, a public hash function $H : \{0,1\}^* \to \mathbb{Z}_q^N$ and user identity *id*. The user identity can be any type of data. For implementation purposes it is considered to be a MAC address, which can be expressed as a 48-bit char array. The random oracle used in the implementation is given as Algorithm 6, using SHA-3 as the hash function [3]. The Extract algorithm outputs the user secret key $\mathbf{SK}_{\mathrm{id}} \in \mathcal{R}_q$. The first steps check if $\mathbf{SK}_{\mathrm{id}}$ is in local storage; if so

Algorithm 2: Extract [11]

```
Data: $\mathbf{B} \in \mathbb{Z}_q^{2N\times 2N}$, $H : \{0,1\}^* \to \mathbb{Z}_q^N$, $id$
Result: $SK_{id}$
1 if $SK_{id}$ is in local storage then
2     return Output $SK_{id}$ to user $id$
3 else
4     $t \leftarrow H(id)$ ;
5     $(s_1, s_2) \leftarrow (t, 0) - Gaussian(B, \sigma, (t, 0))$;
6     $SK_{id} \leftarrow s_2$;
7     return Output $SK_{id}$ to user $id$ and keep in local storage
```

then the secret key has already been extracted for this user so we must use this one. Extracting multiple secret keys for the same user would compromise the security of the system by leaking information. If an existing key is not found, the extraction process begins. Extraction begins by hashing the user *id* to arrive at an integer vector t of length N. The vector t is then concatenated with a zero vector of length N to obtain a $2N$-length vector which will become the centre of the gaussian sample over the lattice. This step uses the **Gaussian Sampler** (see section 7) to sample a vector from the lattice defined by **B**. This vector is then subtracted from the $(t, 0)$ vector to obtain (s_1, s_2) such that $s_1 + s_2 * h = t$.

[3] https://github.com/mjosaarinen/tiny_sha3

This equality is due to the GPV algorithm. The user secret key $\mathbf{SK}_{\text{id}}$ is set to s_2 and this is output and kept in local storage.

The software components of the Extract operation described in Algorithm 2 are expanding the basis into its matrix form, the use of the random oracle, hash function and CSPRNG, the Gaussian Samplers over integer and the lattice and its required Gram-Schmidt Orthogonalisation, and entropy coding. The process as a whole is computationally simple but requires large memory resources. First the private keys (*f, g, F, G*) are used to form the polynomial basis $\mathbf{B}$ and its Gram Schmidt orthogonalisation $\mathbf{B}_{gs}$ (both $2N * 2N$ square matrices). Then the user's unique ID is converted into its public key form (a ring polynomial modulo q) using a random oracle. The proposed software architecture for this is described in Algorithm 6 and relies upon a hash function and a CSPRNG. The final and most intensive stage is obtaining the user secret key using Gaussian Sampling over a lattice.

Gram-Schmidt Orthogonalisation: A Gram-Schmidt Orthogonalised (GSO) basis of the lattice is required by the Gaussian Sampler. In order to accelerate this algorithm, our design currently uses Prest's implementation of the algorithm, which is based on improvements proposed in [20]. This uses $O(2N^2)$ floating point operations as opposed to $O(2N^3)$ of the classical algorithm, for a matrix of dimension $2N$ x $2N$ and considers the isometric structure of the basis: $\{b, r(b), r^2(b), ..., r^{2N-1}(b)\}$ (the cyclic structure of the NTRU basis). The intuition behind this is that if $\tilde{b_k}$ is the GSO of b_k, then $r(\tilde{b_k})$ is the GSO of b_{k+1}. In fact, this is orthogonal to $b_2, ..., b_k$, but not b_1. It is therefore needed to reduce $r(\tilde{b_k})$ with respect to $b_1 - Proj(b_1, Span(b_2, ..., b_k))$.

Obtaining $\mathbf{B}$ and $\mathbf{B}_{gs}$ is relatively straightforward and fast. Depending upon the size of the modulus q it is possible to store B using 16-bit or 32-bit types to both reduce memory storage and improve speed (reduced memory bandwidth, fewer cache misses etc.). To reduce memory usage we have identified that storing $\mathbf{B}$ and $\mathbf{B}_{gs}$ using 32-bit floating-point types is sufficient. Our current implementation uses 32-bit floats to store both $\mathbf{B}$ and $\mathbf{B}_{gs}$ and thus for $n = 512$ we require 4MB for each matrix. We propose that $\mathbf{B}$ could alternatively be computed on-the-fly to further reduce memory usage.

Gaussian Sampler The Gaussian Sampling algorithm given in the paper is a variant of the GPV algorithm [14]. Originally, it was deemed impractical but this distribution of NTRU lattices, along with reducing the standard deviation by a factor of $\sqrt{2}$ due to consideration of Kullbeck-Leibler Divergence (see Section 4 of [11]), improves this. The GPV sampler returns a short lattice vector without revealing the trapdoor. It requires a 1-dimensional Gaussian sampler as a subroutine. A range of algorithms have been presented in the literature for such purposes, for example Bernoulli [12], CDT [21], Ziggurat [7] and Knuth-Yao [17]. This research deploys the CDT method for efficiency purposes. Alternative samplers could also be incorporated within this design and will be considered in future research.

The CDT method gives an efficient form of generating integers according to a Gaussian distribution by reducing the problem to a binary search on precomputed values of the cumulative distribution function. An efficient CDT sampler has been developed in [6] [24]. This CDT method requires 16 kB to store the CDF with 64-bit precision and offers constant-time sampling. A disadvantage of the GPV algorithm is the requirement to sample over varying standard deviations, requiring the re-initialisation of the Gaussian sampler $2N$ times for a ring length of N. We have identified the initialisation time of the Gaussian sampler as a performance bottleneck, and of the range of samplers available to us the CDT method was optimal in this respect. However, we have further modified the sampling scheme to improve performance by reducing the number of re-initialisations from $2N$ to 2. We achieve this by noting that the standard deviation varies insignificantly for the first and latter N samples. This is because it is scaled according to the basis vector modulus, but as this is already capped at $1.17\sqrt{q}$ during basis generation this step has negligible effect. The performance of the Gaussian Sampling algorithm can also be improved by pre-computing the inverse of the norm of the columns of $\mathbf{B}_{gs}$ for a given IBE master key, permitting division to be replaced with faster multiplication.

In terms of side-channel attacks, the Gaussian Sampling algorithm is required only for the server-side operations of Key Generation and Extract. In those applications where the server is vulnerable to physical access by an attacker the constant-time operation of the CDT limits timing analysis.

The GPV samples from a Gaussian distribution with standard deviation that is essentially the length of the Gram-Schmidt Norm. Aside from this, there are no other characteristics of the basis used which could leak information. A better quality (shorter) basis therefore means a narrower Gaussian distribution and the samples are closer to c. The algorithm is a randomised variant of Babai's Nearest Plane [4] for solving (or approaching the solution of) the CVP. Babai's algorithm inductively finds a lattice vector v close to some vector w. To do this, it solves the problem in a lower dimension; specifically, the sub-lattice (or plane) spanned by the first b_{2N-1} basis vectors. This is computed for dimension $2N$ and iterated until dimension 1, when the next "plane" is a vector. The output vectors are summed to "reverse-project" back onto the original lattice. The difference between Babai's algorithm and the GPV sampler is movement to the next plane. Whilst Babai's moves to the nearest plane in each iteration, the GPV sampler chooses the next plane with probability determined by distance to the centre point. The 1-dimensional Gaussian Sampler is used for this plane selection process.

The Gaussian Sampler requires the basis of $2N$-dimensional lattice (the master secret key) $\mathbf{B} \in \mathbb{Z}_q^{2N \times 2N}$, standard deviation $\sigma \gnsim 0$ and centre of the sample $c \in \mathbb{Z}^N$. These input parameters mean the probability of a vector v being sampled is proportional to $exp(-\pi||v-c||^2/2\sigma^2)$. The Gaussian Sampler algorithm outputs a sampled vector v in the Gaussian distribution $\mathcal{D}_{\lambda,\sigma,c}$ over the lattice. The algorithm is presented as Algorithm 3 and is as follows; it iterates through vectors $v_{2N} \to v_0$ and $c_{2N} \to c_0$. The vector v_0 is the sample vector output at the end.

Algorithm 3: Gaussian Sampler [11]

```
  Data: $\mathbf{B}$, $\sigma > 0$, $c \in \mathbb{Z}^n$
  Result: $v \in \mathcal{D}_{\lambda,\sigma,c}$
1 $v_{2N} \leftarrow 0$
2 $c_{2N} \leftarrow 0$
3 for $i \leftarrow 2N, ..., 1$ do
4     $c'_i \leftarrow \langle c_i, \tilde{b_i} \rangle / ||\tilde{b_i}||^2$
5     $\sigma'_i \leftarrow \sigma / ||\tilde{b_i}||$
6     $z_i \leftarrow Gaussian(\sigma'_i, c'_i)$
7     $c_{i-1} \leftarrow c_i - z_i b_i$ and $v_{i-1} \leftarrow v_i + z_i b_i$
8 return $v_0$
```

To begin, v_{2N} is set to the zero vector and c_{2N} is set to the centre vector c. The algorithm then iterates through i from $i = 2N$ to 1. The projection coefficient of c_i on lower-dimensional plane is computed as $c'_i = \langle c_i, \tilde{b_i} \rangle / ||\tilde{b_i}||^2$, where $\tilde{b_i}$ is the Gram-Schmidt orthogonalisation of the basis vector b_i. In Step 5, the standard deviation is scaled down as $\sigma'_i = \sigma / ||\tilde{b_i}||$. Step 6 calls the Gaussian Sampler over the uniform distribution of integers for each i to obtain $z_i = Gaussian(\sigma'_i, c'_i)$. The next steps involve the projection of the centre and sample vector onto the next plane. The vector v_{i-1} is the closest vector to the centre in that (randomly sampled) plane: $c_{i-1} = c_i - z_i b_i$ and $v_{i-1} = v_i + z_i b_i$. Finally, at the finish of all the loops, the vector v_0 is output. For a small cost to RAM requirements it is beneficial to store the inverse of $||\tilde{b_i}||$ (see step 4 and 5 of Algorithm 3). The Gaussian Sampler requires repeated division by this norm and its squared value, the performance of which is much improved by precomputing the inverse value and replacing division with faster multiplication. The Gaussian Sampler is also initialised for each sample that must be produced when generating a user secret key; this is quite intensive and can potentially be omitted if the standard deviation does not vary between iterations of the algorithm. In this design, software acceleration of the Extract process is achieved using auto-vectorisation and more efficient use of types within the dot product and other loops in the Gaussian sampler over the lattice. Once the user secret key has been obtained it can be further compressed for storage or transmission. For this, we use Huffman coding.

3.3 Encryption and Decryption

The Encryption and Decryption algorithms of the scheme are given as Algorithms 4 and 5 respectively. These are based on the original R-LWE cryptosystem [18] and are consequently well studied and refined throughout numerous optimisations.

The Key Generation and Extract components are server-side functions in IBE, whereas Encryption and Decryption are seen as client-side functions and could therefore be implemented in either software or hardware. Therefore we propose the hardware design of [15] could be incorporated with this software design of KeyGen and Extract to create an even faster scheme.

The software procedures needed here are the use of two random oracles $H(id)$ and $H'(k)$, and for each of these, a hash function and CSPRNG are needed. NTT and entropy coding is utilised again.

Algorithm 4: Encrypt [11]

```
  Data: $h, id, m, H, H'$.
  Result: $(u, v, c)$
1 $r, e_1, e_2 \leftarrow \{-1, 0, 1\}^N$ , $k \leftarrow \{0, 1\}$ ;
2 $t \leftarrow H(id)$ ;
3 $u \leftarrow r * h + e_1 \in \mathcal{R}_q$;
4 $v \leftarrow r * t + e_2 + \lfloor q/2 \rfloor$ ;
5 Drop the least significant bits of $v : v \leftarrow 2^l \lfloor v/2^l \rfloor$ ;
6 return $(u, v, m \oplus H'(k))$
```

Algorithm 5: Decrypt [11]

```
  Data: $SK_{id}$, $(u, v, c)$
  Result: $m \in \{0, 1\}^m$
1 $w \leftarrow v - u * s_2$ ;
2 $k \leftarrow \lfloor \frac{w}{q/2} \rfloor$ ;
3 return $m \leftarrow c \oplus H'(k)$
```

Encrypt requires two NTT's and two inverse NTT's in order to efficiently perform a number of ring modular multiplications (for smaller moduli, sparse multiplication can be used), while Decrypt requires only a single NTT and inverse NTT. As the master public key and user secret key are repeatedly used for this purpose their NTT representation is precomputed to reduce complexity at the expense of additional storage. Additionally, the Encrypt operation requires two Random Oracles ($H(id)$ and $H'(k)$) while Decrypt requires one ($H'(k)$). How this operation is constructed is not specified in the original work and is implementation dependent. In Algorithm 6 we describe our method for mapping an arbitrary length user ID into a ring polynomial and in Algorithm 7 we describe a similar process where a random bit string is used to generate a one-time pad. These mapping processes both use the hash function SHA-3 and a NIST AES CTR-DRBG. Encrypt benefits greatly from sparse multiplication when calculating $e_3 * h$ and $e_3 * H(id)$ when q is less than 26 bits, but requires less efficient and aggressive NTT multiplication with larger moduli. Decryption is reliable with NTT with less aggressive reduction when q is less than 26 bits, otherwise the more aggressive and costly version needs to be used, as in Encrypt. Additionally, the proposed architecture of Encrypt and Decrypt has been modified to support any length m of the message instead of specifically N bits, at the cost of performance, but it is envisaged that the increased flexibility is more suitable for full scale testing of the practicality of the scheme.

Algorithm 6: Random Oracle H(id) - convert the ID into a unique polynomial

```
   Data: $id, N, q$
   Result: $t \in \mathcal{R}_q$
1  $s \leftarrow H(id)$
2  $t \leftarrow$ CSRNG$(s) \in \mathbb{Z}_q^N$
3  return $t$
```

Algorithm 7: Random Oracle $H'(k)$ - create a byte stream to use as a one-time pad

```
   Data: $k, N, q$
   Result: $t \in \{0,1\}^m$
1  $s \leftarrow H'(k)$
2  $t \leftarrow$ CSRNG$(s) \in \{0,1\}^m$
3  return $t$
```

3.4 Parameters

One of the main problems surrounding lattice-based cryptography is choosing secure yet efficient parameters. This is due to constantly evolving attacks and the use of bounds rather than concrete estimates in their analysis. The original authors suggest parameters in [11] and these are given in Table 2. The security levels of these parameters are estimated in Section 5.2 of [11] by considering both key recovery and ciphertext attacks. The encryption component of the scheme is the most vulnerable, so the security level estimations depend on the strongest attack known to recover the small errors e_1 and e_2, which is estimated to reach a root Hermite factor of $\gamma = 1.0075$ for $N = 512$ and $\gamma = 1.0044$ for $N = 1024$. However, due to the use of the NTT in this implementation, the value of N has been changed. To apply the NTT, an $2N^{th}$ root of unity has to be found. We also need the condition $q = 1 \mod 2N$ to be satisfied in order to compute the negative wrapped convolution. Therefore, one of the contributions of this design is the proposal of new parameter sets, given in Table 3. We include a parameter l from the Encryption algorithm, which corresponds to the truncation of the ciphertext vector v. The security level is estimated from the root Hermite factor

γ introduced in 2008 [13], which measures the hardness of the underlying lattice problem. For $\gamma \approx 1.004$, we can estimate 80-bit security and for $\gamma < 1.007$, we can estimate 192-bit security. The bit size information for the selected parameter sets is set out in Figure 4. Compression uses Huffman coding throughout. The master public key cannot be compressed as the nature of the NTRU assumption requires it to appear random.

Security parameter	**80-bit**	**192-bit**
Polynomial degree N	512	1024
Modulus q	2^{23} or 2^{24}	2^{27}

Table 2. IBE Scheme original parameters

Parameter Set	**Root Hermite Factor**	**Security Level**	**q**	**q in bits**	**N**	**l**	**2Nth root of unity**
0	1.0075	80	5767169	23	512	19	971
1	1.0079	80	10223617	24	512	20	3981
2	1.0085	<80	51750913	26	512	23	115658
3	1.0038	192	5767169	23	1024	18	19484
4	1.0039	192	10223617	24	1024	20	6877
5	1.0043	<192	51750913	26	1024	22	36945

Table 3. Proposed IBE Scheme parameters

Parameter Set	**Message length**	**Master Public Key**	**Master Private Key**		**User Secret Key**		**Message**
			Uncoded	**Compressed**	**Uncoded**	**Compressed**	
0	512	11776	27648	23650	9216	7576	14336
1	512	12288	27648	23779	9216	7807	14848
2	512	13312	29696	25908	9728	8528	15360
3	1024	23552	51200	43834	17408	15522	29696
4	1024	24576	51200	44161	17408	15702	29696
5	1024	26624	59392	51458	19456	17614	31744

Table 4. DLP IBE key and encrypted message bit sizes. All figures are in bits.

3.5 Results

Figures obtained on an Intel Core i7-6700 CPU at 4.0 GHz are shown in Table 5. This design precomputes $\mathbf{B}$, $\mathbf{B}_{gs}$, the inverse of $||\mathbf{B}_{gs}||$ and the NTT representations of the public master key and the user secret key. The random oracles use the SHA-3 hash function and the AES CTR-DRBG random number generator. A CDT Gaussian Sampler is used to randomly sample over the lattices. It should be noted that this performs all necessary tasks and does not require any offline computation. As the Key Generation is only run once per scheme setup, the time in seconds for one run-through is given. The remaining components are called multiple times (Extract once per user, Encrypt/Decrypt once per message) and so we give the number of times each can be run per second. Extract I refers to the original Extract function, while Extract II utilises compression techniques.

Parameter Set	q	N	KeyGen	Extract I	Extract II	Encrypt	Decrypt
0	5767169	512	2.666	544	521	9726	33070
1	10223617	512	2.74	541	526	9753	33359
2	51750913	512	4.034	537	527	9390	22489
3	5767169	1024	16.860	138	135	4179	17493
4	10223617	1024	23.004	137	133	3854	17568
5	51750913	1024	25.126	137	134	3598	11526

Table 5. DLP IBE performance in terms of KeyGen processing time in seconds and Extract, Encrypt and Verify operations per second on an Intel Core i7 6700 @ 4 GHz with SHA-3, CDT Gaussian Sampling and AES CTR-DRBG

For reference, we now give comparable figures from Prest's NTL-based implementation in Table 6 (scaled up to account for differences in CPU). Prest's results were given in ms on an Intel Core i5-3210M laptop with 2.5GHz CPU and 6GB RAM so we have converted to "per second" to represent how they would look like on our 4GHz platform: For example, 8.6ms = 0.0086s, which means 116.28 per sec at 2.5GHz, which is equivalent to 116.38/2.5 * 4 =186.04 per sec at 4GHz. We compare Prest's 80-bit and 192-bit security results to our parameter sets 2 and 5 respectively, without compression. It can be seen that the proposed software architecture outperforms the original PoC in all respects, for example for $N = 512, q = 2^{24}$ (or 80-bit security) with NTL, the Extract function can be run almost three times as many per second, and the Encrypt/Decrypt over x5 and x12 times respectively. Note that the Master KeyGen timings are not specified here, but as this is only run once per scheme set-up it can be temporarily disregarded.

In comparison to classical IBE schemes, the DLP lattice-based IBE scheme also has respectable performance. In 2011, performance testing of the Boneh-Franklin IBE scheme [8] on a Pentium Dual T2330 at 1.60GHz reported that Extraction could be run at 170.6 ops/s, Encrypt at 1.08 op/s and Decrypt at

Security Level	Extract		Encrypt		Decrypt	
	This work	**Prest**	**This work**	**Prest**	**This work**	**Prest**
80-bit	537	186	9725	1758	33070	2580
192-bit	137	49	3598	856	11526	1260

Table 6. Prest's NTL implementation results comparison, operations per sec

1.26 op/s. Therefore, this research shows that replacing current schemes with post-quantum schemes will improve security without impacting efficiency.

4 Conclusions and Further Research

The research presented here demonstrates how a lattice-based IBE scheme performs on software. It can be used as a benchmark for further improvements within the scheme and provides a starting point for further investigation. We have proposed the first working, efficient C implementation of the DLP-IBE scheme with a range of novel software optimisations to enhance performance and have discovered many areas for potential optimisations for a range of targeted devices. The future aim is to consider suitable client-side optimisations for a range of constrained devices, such as those likely to be encountered in IoT, as well as a range of server-side optimisations, such as GPU and multithreading. We intend to carry out a full performance testing of several aspects of the scheme. The Gaussian Sampler is a main bottleneck of the scheme. There is scope to investigate other variants of the GPV sampler, by computing the memory-heavy Gram-Schmidt orthogonalisation on the fly or further acceleration by properties of the NTRU basis structure. We also intend to test different hash functions and Extended Euclidean Algorithms to evaluate the effect on the scheme. Lastly, but perhaps most importantly, is the choice of parameters for the scheme. Currently, we consider 80-bit and 192-bit security levels but it could be insightful to test parameter sets for higher security and determine how they would fare on small devices.

Additionally, the implementation of the scheme opens up other applications for investigation and further research. A hash-and-sign digital signature scheme can use the components of the IBE scheme in a different way. The public verification key corresponds to the master public key, the secret signing key corresponds to the master secret key, messages replace user IDs and signatures replace user secret keys. Secondly, an Authenticated Key Exchange (AKE) scheme can be constructed. This consists of a Key Encapsulation Mechanism (KEM) together with a digital signature scheme. Therefore we can use Pino et al. [22] KEM (based on NTRU) with this digital signature scheme to form a AKE scheme. Both these additional schemes offer the hardness properties and quantum-resilience of lattice-based primitives.

5 Acknowledgements

The authors would like to thank the anonymous reviewers of SAC 2017 and IMACC 2017 for their careful reading of the paper and their diligent comments. The authors would also like to acknowledge that this work was partially funded by the European Union H2020 SAFEcrypto project (grant no. 644729).

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (h)ibe in the standard model. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 553–572. Springer (2010)
2. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: STOC '97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. pp. 284–293 (1997)
3. Apon, D., Fan, X., Liu, F.H.: Compact identity based encryption from lwe. IACR Cryptology ePrint Archive (2016)
4. Babai, L.: On Lovasz lattice reduction and the nearest lattice point problem. Combinatorica 6(1), 1–13 (1986)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Annual International Cryptology Conference. pp. 213–229. Springer (2001)
6. Brannigan, S., Smyth, N., Oder, T., Valencia, F., O'Sullivan, E., Güneysu, T., Regazzoni, F.: An investigation of sources of randomness within discrete Gaussian sampling. Cryptology ePrint Archive, Report 2017/298 (2017), http://eprint.iacr.org/2017/298
7. Buchmann, J., Cabarcas, D., Göpfert, F., Hülsing, A., Weiden, P.: Discrete ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers. In: International Conference on Selected Areas in Cryptography. pp. 402–417. Springer (2013)
8. Cheng, P., Gu, Y., Lv, Z., Wang, J., Zhu, W., Chen, Z., Huang, J.: A performance analysis of identity-based encryption schemes. In: INTRUST. pp. 289–303. Springer (2011)
9. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: A Homomorphic LWE Based E-voting Scheme, pp. 245–265. Post-Quantum Cryptography, Springer (2016)
10. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: IMA International Conference on Cryptography and Coding. pp. 360–363. Springer (2001)
11. Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices, pp. 22–41. Advances in Cryptology ASIACRYPT 2014, Springer (2014)
12. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians, pp. 40–56. Advances in Cryptology CRYPTO 2013, Springer (2013)
13. Gama, N., Nguyen, P.: Predicting lattice reduction. Advances in Cryptology EUROCRYPT 2008 pp. 31–51 (2008)
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions (2007), http://eprint.iacr.org/

15. Gúneysu, T., Oder, T.: Towards lightweight identity-based encryption for the post-quantum-secure internet of things. In: 2017 18th International Symposium on Quality Electronic Design (ISQED). pp. 319–324 (2017), iD: 1
16. Katsumata, S., Yamada, S.: Partitioning via non-linear polynomial functions: More compact ibes from ideal lattices and bilinear maps. In: Advances in Cryptology ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22. pp. 682–712. Springer (2016)
17. Knuth, D.E., Yao, A.C.: The complexity of nonuniform random number generation. Algorithms and complexity: new directions and recent results pp. 357–428 (1976)
18. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. Journal of the ACM (JACM) 60(6), 43:1–43:35 (2013)
19. Lyubashevsky, V.: Lattice signatures without trapdoors, pp. 738–755. Advances in Cryptology, EUROCRYPT 2012, Springer (2012)
20. Lyubashevsky, V., Prest, T.: Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and Gaussian sampling in structured lattices, pp. 789–815. Advances in Cryptology–EUROCRYPT 2015, Springer (2015)
21. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Annual Cryptology Conference. pp. 80–97. Springer (2010)
22. Pino, R.D., Lyubashevsky, V., Pointcheval, D.: The whole is less than the sum of its parts: Constructing more efficient lattice-based AKEs. In: International Conference on Security and Cryptography for Networks. pp. 273–291. Springer (2016)
23. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) 56(6), 34:1–34:40 (2009)
24. Saarinen, M.J.O.: Arithmetic coding and blinding countermeasures for lattice signatures. Cryptology ePrint Archive, Report 2016/276 (2016), http://eprint.iacr.org/2016/276
25. Sakai, R., Kasahara, M.: ID based cryptosystems with pairing on elliptic curve. IACR Cryptology ePrint Archive 2003, 54 (2003)
26. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Workshop on the Theory and Application of Cryptographic Techniques. pp. 47–53. Springer (1984)
27. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review 41(2), 303–332 (1999)
28. Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 27–47. Springer (2011)
29. Yamada, S.: Adaptively secure identity-based encryption from lattices with asymptotically shorter public parameters. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 32–62. Springer (2016)