Practical Applications of Improved Gaussian Sampling for Trapdoor Lattices

Kamil D. Gür*, Yuriy Polyakov*‡, Kurt Rohloff*‡, Gerard W. Ryan*, Hadi Sajjadpour* and Erkay Savaş*†

* NJIT Cybersecurity Research Center
New Jersey Institute of Technology, Newark, NJ, USA 07102
Email: {kg365,polyakov,rohloff,gwryan,ss2959}@njit.edu
† Sabancı University, Tuzla, Istanbul, Turkey 34956
Email: erkays@sabanciuniv.edu
‡ Corresponding Authors

Abstract

Lattice trapdoors are an important primitive used in a wide range of cryptographic protocols, such as identity-based encryption (IBE), attribute-based encryption, functional encryption, and program obfuscation. In this paper, we present software implementations of the Gentry-Peikert-Vaikuntanathan (GPV) digital signature, IBE and ciphertext-policy attribute-based encryption (CP-ABE) schemes based on an efficient Gaussian sampling algorithm for trapdoor lattices, and demonstrate that these three important cryptographic protocols are practical. One important aspect of our implementation is that it supports prime moduli, which are required in many cryptographic schemes. Also, our implementation uses bases larger than two for the gadget matrix whereas most previous implementations use the binary base. We show that the use of higher bases significantly decreases execution times and storage requirements. We adapt IBE and CP-ABE schemes originally based on learning with errors (LWE) hardness assumptions to a more efficient Ring LWE (RLWE) construction. To the best of our knowledge, ours are the first implementations employing the Gaussian sampling for non-binary bases of the gadget matrix. The experimental results demonstrate that our lattice-based signature, IBE and CP-ABE implementations, which are based on standard assumptions with post-quantum security, provide a performance comparable to the recent state-of-the-art implementation works based on stronger/non-post-quantum assumptions.

Keywords: lattice-based cryptography · RLWE · identity-based encryption · attribute-based encryption · GPV digital signature · Gaussian sampler

I. Introduction

Lattice-based cryptography [47], [49], [51], a recent but increasingly important family of cryptographic systems, becomes a center of attraction in academia as lattice-based cryptographic schemes are generally believed to be "post-quantum" in the sense that they are secure against quantum computing attacks [50]. Also, lattice-based cryptography supports homomorphic encryption [13], [23], [28] and is used in the construction of many advanced cryptographic schemes, such as identity-based encryption (IBE) [10], attribute-based encryption (ABE) [20], [54], predicate encryption (PE) [29], and software obfuscation [14].

Many lattice-based cryptographic schemes rely on the hardness assumptions of learning with errors (LWE) [52] or the more efficient ring learning with errors (RLWE) problems [39], [40]. Another related concept is strong lattice trapdoors, which involve sampling from an n-dimensional lattice $\mathcal{L}$ with a Gaussian-like distribution [27], hence the name *Gaussian sampling*. Lattice trapdoors are needed to implement advanced cryptographic algorithms, such as IBE, ABE [20], PE, and conjunction obfuscation [18].

Quite a few theoretical works outline actual construction techniques and explain in detail as to how these trapdoors are efficiently and securely constructed [25], [27], [42] whereas there are only very few attempts to report on actual implementations. Bansarkhani and Buchmann [8] implement two classes of trapdoors that work in matrix and ring settings, respectively, and conclude that the ring-based Gaussian sampler is more efficient than the matrix version from both execution and storage requirement points of view. The Gaussian samplers are used to implement the GPV signature scheme [27], which yields timings almost comparable to conventional (non-post-quantum) signature schemes [8]. Nevertheless, the ring-based Gaussian sampler works only with a power of two modulus, which severely limits its applicability to more involved cryptographic schemes that usually require prime (or arbitrary) moduli.

Genise and Micciancio [25] present an efficient Gaussian sampling method for arbitrary moduli, which is implemented by Gur *et al.* [32]. The implementation of the signature scheme with an arbitrarily chosen prime modulus [32] proves to be faster and requires less memory than the implementation of the signature scheme by Bansarkhani and Buchmann [8]. While these works provide invaluable insights, more research into the subject is urgently needed to assess the practicality of the Gaussian sampling methods for more involved cryptographic schemes. This work is the first attempt in this direction to show that cryptographic schemes, such as IBE and ABE, can be efficiently implemented using Gaussian sampling for lattice trapdoors.

Identity-based encryption (IBE) [10] is a public-key cryptography (PKC) scheme, in which an arbitrary string that uniquely identifies a party/individual can be used as her public key. IBE can be utilized to help eliminate or simplify unduly complicated public-key infrastructures for managing certificates. To the best of our knowledge, the works [22] and [41] are the only studies that report on IBE implementations based on lattice trapdoors.

Attribute-Based Encryption (ABE), which is usually considered as a generalization of IBE [9], [31], [54], is also a PKC scheme, which enables the decryption of a ciphertext by a user only if a certain access policy defined over a set of attributes is satisfied by the user (or more precisely by her attributes). Besides helping to build complex access control systems, ABE is proposed for implementing other interesting applications such as audit log encryption and targeted/broadcast encryption [31].

ABE has two main flavors of constructions: Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE). CP-ABE has been more widely studied and implemented in the literature [9], [21], [56], [57], [59]. In CP-ABE, the ciphertext is encrypted under an access policy, and a user private key for decryption is generated for the set of attributes held by the user. In KP-ABE [31], [45], [54], on the other hand, the message is encrypted using the attribute values as public keys, and a secret key is generated for a particular access policy defined over the set of attributes.

Two classes of cryptographic primitives are generally used in the construction of ABE schemes: bilinear pairings and lattices. The majority of ABE schemes are based on bilinear pairings [10], such as [30], [31], [36], [37], [56]. Software implementations of pairing-based ABE constructions are reported in works [9], [55], [57]. To the best of our knowledge, this study is the first that implements a lattice-based CP-ABE scheme using a Gaussian sampler.

A. Our Contribution

In this work, we develop optimized (RLWE) variants of the IBE [27] and CP-ABE [58] schemes originally formulated based on LWE. Our optimized variants have significantly lower (often more than by one order of magnitude) computational and storage complexity.

We implement the trapdoor sampling algorithm proposed by Genise and Micciancio [25] for a gadget matrix (lattice) with an arbitrary base (in constrast to the binary-base implementation [32]). Using a larger base (up to a certain limit) significantly improves the performance of all trapdoor-dependent operations. For instance, the runtime of trapdoor sampling gets improved by up to 5x.

We implement the ring-based GPV signature, IBE, and CP-ABE schemes in PALISADE based on the efficient trapdoor sampling and evaluate their performance. Our analytical and experimental results show that the use of generalized gadget lattices substantially improves the overall performance of all three schemes.

B. Related Work

After [18], [32], this paper is the third that reports on the implementation of the efficient Gaussian sampling method for lattice trapdoors proposed by Genise and Micciancio [25], which works with arbitrary moduli. Gur *et al.* [32] use a $\mathbf{G}$-lattice (gadget lattice or matrix) constructed by the primitive vector $\mathbf{g}^T = \{2^0, 2^1, 2^2, \ldots, 2^k - 1\}$, where $k = \lceil \log_2 q \rceil$ and q is the modulus. Our implementation utilizes generalized $\mathbf{G}$-lattice with the vectors $\mathbf{g}^T = \{b^0, b^1, b^2, \ldots, b^k - 1\}$ and works with any base $b \geq 2$. We demonstrate that using relatively larger bases for the $\mathbf{G}$-lattice improves the execution times and storage requirements significantly. Similar to this work, Cousins *et al.* [18] also use the trapdoor construction with generalized $\mathbf{G}$-lattice with prime moduli and large bases, but for a different application (cryptographic program obfuscation).

Two closely related previous works [8] and [32] report only on the performance of GPV digital signature algorithm. In this work, we not only demonstrate that our new implementation of Gaussian sampling for lattice trapdoors significantly improves both execution times and storage requirements of GPV signature, but we also implement lattice-based IBE and CP-ABE schemes. For the latter categories of cryptographic algorithms, we adapt the IBE and CP-ABE schemes based on LWE hardness assumptions to the RLWE setting for efficient implementation. We show that our IBE construction is IND-CPA secure whereas the CP-ABE construction is secure against selective chosen plaintext attack (sCPA).

To the best of our knowledge, ours is the third IBE implementation based on lattice trapdoors (and the first using Gaussian sampling for lattice trapdoors) in the literature whereas the CP-ABE implementation is the first. We demonstrate that both schemes, which are based on standard assumptions with post-quantum security, provide a performance comparable to the recent state-of-the-art implementation works, which are based on stronger/non-post-quantum assumptions.

C. Organization

The rest of the paper is organized as follows: We provide the necessary background information in Section II. The Gaussian sampling algorithm for lattice trapdoors is explained in Section III. We explain the GPV signature, the RLWE-based IBE and CP-ABE schemes and give proofs for their correctness and security constraints in Sections IV-A, IV-B and IV-C, respectively. Section V provides the implementation details and results such as execution times and storage requirements, including a comparison with similar works in the literature. Section VI concludes the paper.

II. Preliminaries

In this section, we provide mathematical background and introduce the security assumptions used in the paper.

A. Mathematical Notations And Definitions

Let $\mathcal{R} = \mathbb{Z}[x]/\left\langle x^n+1\right\rangle$ be a cyclotomic polynomial ring where the ring elements are polynomials of at most degree $n-1$ with integer coefficients and n is a power of 2. And let also $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ be a ring where the arithmetic operations on polynomial coefficients are performed modulo q and coefficients are represented as integers in the interval $\left(\left\lfloor -\frac{q}{2}\right\rfloor, \left\lfloor \frac{q}{2}\right\rfloor\right]$. Also, $\mathcal{R}_q^{1\times m}$, $\mathcal{R}_q^{m}$, and $\mathcal{R}_q^{m\times m}$ stand for a row vector, column vector and matrix of ring elements in R_q, respectively, for an integer $m > 1$.

Throughout the paper, boldface letters always denote matrices and vectors (e.g., $\mathbf{a} = (a_0,\ a_1,\ \ldots,\ a_{n-1})$). While a polynomial in $\mathcal{R}_q$ can be represented as a vector in $\mathbb{Z}_q^n$ $(= (\mathbb{Z}/q\mathbb{Z})^n)$, an integer coefficient of a polynomial can be represented as a vector of digits in base b.

We also denote the infinity norm of a polynomial or a vector as $||\cdot||_\infty$ (only *the norm* and $||\cdot||$ for simplicity). A polynomial or a vector is *short* if its norm is small.

B. Efficient Arithmetic in $\mathcal{R}_q$

For arithmetic in cyclotomic polynomial rings, we rely on the number theoretic transform (NTT) [17], which is a special form of discrete Fourier transform defined over finite fields or rings. As reduction with $x^n + 1$ is very easy (since $x^n = -1$), it can be incorporated into NTT operations; resulting in a technique, which is known as *negative wrapped convolution* [16]. The method utilizes a primitive $2n$-th root of unity ζ that exists if $q \equiv 1 \pmod{2n}$.

When a ring element $a \in \mathcal{R}_q$ (in polynomial representation) is transformed into $\tilde{\mathbf{a}}$ using NTT, the latter is said to be in the *evaluation* representation, whereby the multiplication is extremely efficient as it is performed element-wise. The transformation operations themselves (NTT and inverse NTT) are usually the computational bottlenecks. Therefore, provided that the cryptographic computations permit, it is better to keep operands in the evaluation representation as long as possible; an approach adopted in this paper to accelerate cryptographic computations.

C. Lattices

A full rank lattice Λ, which is a discrete additive subgroup of the n-dimensional real space $\mathbb{R}^n$, is the integer span $\Lambda = \mathcal{L}(\mathbf{B}) = \{\mathbf{Bz} = \sum_{i=1}^{n} z_i\mathbf{b}_i | \mathbf{z} \in \mathbb{Z}^n\}$ of a basis $\mathbf{B} = (b_1, \ldots, b_n) \subseteq \mathbb{R}^n$. The *minimum distance* $\lambda_1(\Lambda)$ of a lattice Λ is the length (usually the Euclidean ℓ_2 norm) of its shortest nonzero vector; namely $\lambda_1(\Lambda) = \min_{0 \neq \mathbf{x} \in \Lambda} ||\mathbf{x}||$. Informally speaking, we can define two hard computational problems on lattices

Definition 2.1: **Shortest Vector Problem** (SVP) Given a lattice basis $\mathbf{B}$ for Λ, find the shortest nonzero vector in Λ.

Definition 2.2: **Shortest Independent Vectors Problem** (SIVP) Given a lattice basis $\mathbf{B} \in \mathbb{Z}^{n \times n}$, find n linearly independent lattice vectors $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_n)$, where $\mathbf{s}_i \in \Lambda$, which minimizes the maximum of the infinity norms of $\mathbf{s}_i$ for $i = 1, \ldots n$.

One can also consider their approximation variants; for example, $\mathtt{SVP}_\gamma$, where the goal is to find a short vector whose norm is at most $\gamma\lambda_1(\Lambda)$ for a given factor γ.

q-ary lattices is an important category of lattices which finds wide use in lattice-based cryptography. Given a uniformly randomly chosen matrix of $\mathbf{A} \in \mathbb{Z}^{n \times m}$ for some integers n, m, q we can define two q-ary lattices,

$$\begin{aligned}\Lambda_q(\mathbf{A}) =& \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} = \mathbf{A}^T\mathbf{s} \pmod q \text{ for some } \mathbf{s} \in \mathbb{Z}^n\} \\ \Lambda_q^\perp(\mathbf{A}) =& \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{A}\mathbf{y} = 0 \pmod q\}.\end{aligned}$$

Finding short vectors in q-ary lattices is shown to be as hard as the approximate variant of certain lattice problems (e.g. SIVP) [1], [27], [43]. One such problem is the shortest integer solution (SIS) problem introduced and analyzed by Ajtai [2].

Definition 2.3: **Shortest Integer Solution** (SIS) Given n, m, q, $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a norm bound $1 \leq \nu < q$, find $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A})$ with $0 < ||\mathbf{v}||_2 \leq \nu$.

$D_{\Lambda,\mathbf{c},\sigma}$ is used to denote the n-th dimensional Gaussian distribution over a lattice $\Lambda \in \mathbb{R}^n$, where $\mathbf{c} \in \mathbb{R}^n$ is the center and $\sigma \in \mathbb{R}$ is the distribution parameter. Gaussian lattice sampling denoted as $\mathbf{x} \leftarrow D_{\Lambda,\mathbf{c},\sigma}$ assigns the probability $\rho(\mathbf{x}) / \sum_{\mathbf{z} \in \Lambda} \rho_{\mathbf{c},\sigma}(\mathbf{z})$ for $\mathbf{x} \in \Lambda$, where $\rho = \mathtt{exp}(-\pi||\mathbf{x} - \mathbf{c}||/\sigma^2)$. When omitted, the distribution (or smoothing) parameter and the center are taken to be 1.0 and $\mathbf{0}$, respectively. At a more basic level, $e \leftarrow D_{\mathbb{Z}^n,\mathbf{c},\sigma}$ denotes the sampling of n independent integers with $\mathbf{c} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$.

The notation $a \leftarrow_U \mathbb{Z}_q$ (or $a \leftarrow_U \mathbb{Z}_q^n$, $a \leftarrow_U \mathcal{R}_q$) is used for sampling from a discrete uniformly random distribution.

D. Ring Learning with Errors

Ring learning with errors (RLWE) problem, whose hardness can be based on the worst-case hardness of ideal lattice problems (due to quantum reduction from worst-case approximate SVP on ideal lattices to the search version of RLWE [39]), can be defined in the context of cyclotomic polynomial rings $\mathcal{R}_q = \mathbb{Z}_q / \langle x^n + 1 \rangle$, where q is prime and n is a power of two.

Let s be a random (and unknown) polynomial in $\mathcal{R}_q$. We consider a number of pairs of the form $(a_i, a_i s + e_i) \in \mathcal{R}_q^2$, where a_i stands for uniformly randomly chosen polynomials in $\mathcal{R}_q$ and $e_i \leftarrow D_{\mathcal{R},\sigma}$ with a relatively small $\sigma \in \mathbb{R}$. Now, we can give RLWE hardness assumptions that we use to prove the security of cryptographic algorithms presented in this paper.

Definition 2.4: **Search RLWE assumption** is that it is hard to find s given a list of pairs $(a_i,\ a_i s + e_i)$ for $i = 0, \ldots, t$.

Definition 2.5: **Decision RLWE assumption** is that it is hard to distinguish between polynomials $(a_i s + e_i)$ and (b_i) for $i = 0, \ldots, t$, where b_i's are uniformly randomly chosen polynomials in $\mathcal{R}_q$.

The hardness of RLWE suggests that polynomials $a_i s + e_i$ appear to be "uniformly random" (more formally, pseudorandom), and it is difficult to distinguish between these RLWE pseudorandom polynomial samples and true uniformly random polynomials.

Informally speaking, in both definitions, t stands for the number of samples a polynomial-time adversary or distinguisher can obtain. The hardness of the RLWE assumptions depends on the choice of ring dimension n, the size of q and a bound Δ for the coefficients of e_i, which is determined by the distribution parameter σ of $D_{\mathcal{R},\sigma}$.

For the RLWE hardness assumptions to hold, the values of n and q must be selected properly. While obtaining accurate security estimates for given values of n and q is difficult and requires involved computations and arguments, several pioneering works provide reliable guidelines for this purpose [4]–[6], [26], [34], [38].

In this work, adopting the approach in the white paper [15] to obtain security estimates for a specific choice of parameters, we use the LWE estimator [3] based on the works [4]–[6]. Using the version with commit number

TABLE I
SECURITY ESTIMATES (FOR CLASSICAL/QUANTUM COMPUTERS) WITH $\sigma = 4.578$ AND UNIFORM DISTRIBUTION FOR SECRET KEYS (n IS THE RING DIMENSION, k – MODULUS BITWIDTH, λ – SECURITY PARAMETER, AND δ – ROOT HERMITE FACTOR) [3]

n	k	λ		δ
		classical	quantum	
512	24	84.1	79.1	1.006546
1024	27	149.3	138.3	1.003941
1024	31	126.2	117.4	1.004571
1024	32	121.6	113.1	1.004727
1024	33	117.2	109.2	1.004886
1024	34	113.1	105.4	1.005045
1024	35	109.0	101.7	1.005204
1024	36	105.5	98.5	1.005630
1024	37	102.5	95.9	1.005514
1024	38	99.1	92.7	1.005678
1024	39	96.1	90.1	1.005837

$cc5f6e8$, we list the security estimates for parameter sets used in this paper in Table I. The security estimator provides estimates for both classical and quantum computers considering three different attack types, namely i) the unique shortest vector attack (uSVP), ii) the decoding attack, and iii) the dual attack. In Table I, we list the most conservative security estimates for given combinations of ring dimension and modulus size. Naturally, one should take the minimum of the estimates in the table for a specific choice of parameters if post-quantum security is targeted. Readers are referred to [4]–[6], [15], [38] for more information about the attacks.

When working with discrete Gaussian distribution to approximate the continuous Gaussian distribution, one needs to introduce a smoothing parameter $\eta_\epsilon(\Lambda)$ that can be estimated as $\eta_\epsilon(\Lambda) \leq \|\bar{\mathbf{B}}\| \cdot r$ (Lemma 3.1 in [27]), where $\|\bar{\mathbf{B}}\|$ is the norm of an ortogonalized lattice basis $\bar{\mathbf{B}}$, $r \approx \sqrt{\ln(2n_m/\epsilon)/\pi}$, n_m is the maximum ring dimension, and ϵ is the bound on the statistical error introduced by each randomized-rounding operation. For $n_m \leq 2^{14}$ and $\epsilon \geq 2^{-80}$, the value of $r \approx 4.578$. In all of our procedures, the distribution parameter for discrete Gaussian distribution has to be set to at least r. For simplicity of notation, we reuse σ as the distribution parameter for discrete Gaussian distribution and set it to 4.578 in the rest of the paper whenever we deal with standard zero-centered Gaussian integer sampling (i.e., $\eta_\epsilon(\mathbb{Z}^n) \leq r$ per Lemma 2.5 of [46]).

The concept of trapdoors is well-known in cryptographic context, whereby trapdoor is an extra piece of information that enables to efficiently compute a solution to a hard problem. In this paper, we rely on the lattice trapdoors introduced in [42]. Let $\mathbf{A} \in \mathcal{R}_q^{1\times m}$ be a uniformly randomly selected vector of ring elements. Informally speaking, for an arbitrarily chosen $\beta \in \mathcal{R}_q$, it is computationally hard to find a short vector of ring elements $\boldsymbol{\omega} \in \mathcal{R}^m$ that satisfies $\mathbf{A}\boldsymbol{\omega} = \beta$. Furthermore, the vectors in the solution must be spherically distributed with a Gaussian function and a distribution parameter σ_s; namely we should have $\boldsymbol{\omega} \leftarrow D_{\Lambda,\sigma_s}$.

Finding such short vectors is usually referred as *preimage (Gaussian) sampling* operation for an arbitrary *syndrome* β. The hardness assumption can be based on the hardness of the SIVP$_\gamma$ problem, namely $\texttt{SIVP}_\gamma$. On the other hand, a trapdoor $\mathbf{T_A}$ for $\mathbf{A}$ can be used to compute such short vectors efficiently as will be shown in our construction in Section III.

E. IBE and CP-ABE Basics

In identity-based encryption (IBE) schemes, an arbitrary string ($\texttt{ID}$) that uniquely identifies an individual is used as a public key to encrypt a plaintext while the corresponding private key must be generated by a trusted third party, usually referred as the private key generator (PKG). A hash function is used to transform an identity to an element of the underlying mathematical object, such as a ring element.

IBE schemes consist of four algorithms: *Setup*, *Encryption*, *Key Generation*, and *Decryption*. During the setup, PKG takes security parameter λ and generates a master public and secret key pair: $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \textsc{Setup}(\lambda)$.

In key generation, PKG uses MSK to generate the private key that corresponds to a user identity (IBE): $\boldsymbol{\omega}_{\texttt{ID}} \leftarrow$ $\textsc{KeyGen}(\texttt{ID}, \text{MSK}, \text{MPK})$

Sender uses MPK and $\texttt{ID}$ to encrypt a message μ and obtains the ciphertext $\mathbf{C} \leftarrow \textsc{Encrypt}(\mu, \text{MPK}, \texttt{ID})$. Then, receiver calls $\textsc{Decrypt}(\mathbf{C}, \boldsymbol{\omega}_{\texttt{ID}})$ function to obtain the plaintext message μ. Decryption succeeds if the receiver possesses the correct private key.

In CP-ABE, an access policy defines the rules as to who can decrypt a ciphertext. Therefore, an access policy over a subset of universal set of attributes $\mathcal{X} = \{x_1, x_2, \ldots, x_\ell\}$ serves as a public key during encryption. A private key corresponding to a set of attributes held by a user is generated by PKG[1].

CP-ABE schemes consist of same four algorithms as IBE. During the setup, PKG takes λ and $\mathcal{X}$ as input and generates a master public and secret key pair: $(\text{MPK}, \text{MSK}) \leftarrow \textsc{Setup}(\lambda, \mathcal{X})$. In key generation, PKG uses MSK to generate the private key that corresponds to a subset of attributes held by a user: $\boldsymbol{\omega}_{\mathcal{Y}} \leftarrow \textsc{KeyGen}(\mathcal{Y}, \text{MSK}, \text{MPK})$, where $\mathcal{Y} \subseteq \mathcal{X}$ represents the set of attributes held by the user.

Sender uses MPK and an access policy $\mathcal{W}$ to encrypt a message μ and obtains the ciphertext $\mathbf{C} \leftarrow \textsc{Encrypt}(\mu, \text{MPK}, \mathcal{W})$. An access policy is usually represented as a Boolean expression over a subset of attributes $\mathcal{Z}$, namely $\mathcal{W} = F(\mathcal{Z})$, where $\mathcal{Z} \subseteq \mathcal{X}$. When the set of user attributes $\mathcal{Y}$ satisfies an access policy, we write $\mathcal{Y} \vdash \mathcal{W}$. Then, receiver calls $\textsc{Decrypt}(\mathbf{C}, \boldsymbol{\omega}_{\mathcal{Y}})$ for decryption, which succeeds if the receiver possesses the correct private key, which happens only when his attributes satisfy the access policy used in encryption.

One important property of ABE schemes is that they are *collision resistant* in the sense that the users cannot combine their private keys to decrypt a ciphertext, if their individual attributes do not satisfy the access policy in the ciphertext.

III. Gaussian Sampling Algorithms for Rings

For lattice trapdoor sampling we utilize the ring version of the trapdoor construction examined and implemented in work [8] (depicted in Algorithm 1). In the algorithm, $\bar{m} = \lfloor \log_b(q) + 1 \rfloor$ is the length of modulus q in base b, which can be any integer. In our construction we use only power of two bases for efficiency. The trapdoor consists of two short vectors sampled using a Gaussian distribution with the distribution parameter σ, $\mathbf{T_A} = (\boldsymbol{\rho}, \boldsymbol{\upsilon})$. While the trapdoor $\mathbf{T_A}$ is secret, the public key $\mathbf{A}$ is pseudo-random and enjoys the RLWE hardness assumptions.

The work [8] provides a very efficient preimage sampling algorithm for a power of two modulus. The authors show that the trapdoor can be efficiently used in digital signature algorithms. However, for many other cryptographic schemes, such as IBE and ABE, a prime modulus is more common. Therefore, the preimage sampling algorithm for $\mathbf{G}$-lattices with arbitrary modulus is proposed by Genise and Micciancio [25], which is also used and implemented in this work.

Algorithm 1 Trapdoor generation for RLWE-based schemes [8]

```
function TrapGen($\lambda$)
    Determine $\sigma$, $q$ and $n$ for the security level $\lambda$
    $\bar{m} \leftarrow \lfloor \log_b(q) + 1 \rfloor$
    $a \leftarrow_U \mathcal{R}_q$
    $\boldsymbol{\rho} \leftarrow [\rho_1, \ldots, \rho_{\bar{m}}]$ where $\rho_i \leftarrow D_{\mathcal{R},\sigma}$ for $i = 1, \ldots, \bar{m}$
    $\boldsymbol{\upsilon} \leftarrow [\upsilon_1, \ldots, \upsilon_{\bar{m}}]$ where $\upsilon_i \leftarrow D_{\mathcal{R},\sigma}$ for $i = 1, \ldots, \bar{m}$
    $\mathbf{A} \leftarrow [a, 1, g_1 - (a\rho_1 + \upsilon_1), \ldots, g_{\bar{m}} - (a\rho_{\bar{m}} + \upsilon_{\bar{m}})]$
    return $(\mathbf{A}, \mathbf{T_A} = (\boldsymbol{\rho}, \boldsymbol{\upsilon}))$
end function
```

Using the *primitive* vector $\mathbf{g}^T = (b^0, b^1, \ldots, b^{\bar{m}-1})$, introduced by Micciancio and Peikert [42], we can generate a $\mathbf{G}$-*lattice*, for which preimage sampling can be efficiently computed. If preimage sampling is efficiently computable for the $\mathbf{G}$-lattice, we can show that it is also efficiently computable for the lattice $\mathbf{A}$ given the trapdoor $\mathbf{T_A}$. Namely, for an arbitrary syndrome $\beta \in \mathcal{R}_q$, it is easy to see that $\mathbf{y} = (\mathbf{x}^T\boldsymbol{\rho}, \mathbf{x}^T\boldsymbol{\upsilon}, x_1, x_2, \ldots, x_{\bar{m}})$ is a short solution to $\mathbf{A}\mathbf{y} = \beta$, where $\mathbf{x}$ is a short solution to $\mathbf{g}^T\mathbf{x} = \beta$.

[1]In key-policy ABE (KP-ABE) schemes, the private key corresponds to an access policy. However, KP-ABE is beyond the scope of our paper and the reader is referred to [20] for further information.

However, the framework [27] requires that the preimage sampling algorithm produce a spherically distributed solution for a given syndrome $\beta \in \mathcal{R}_q$. It is shown in works [42] and [8] that solutions in the form of $\mathbf{y} = (\mathbf{x}^T\boldsymbol{\rho}, \mathbf{x}^T\boldsymbol{\upsilon}, x_1, x_2, \ldots, x_{\bar{m}})$ are not spherically distributed, but ellipsoidal, and therefore leak information about the trapdoor. Therefore, a perturbation method is proposed by Micciancio and Peikert [42]. Algorithm 2 gives a high-level description of our secure preimage sampling algorithm. The algorithm relies on the preimage sampling on G-lattices, but it perturbs the preimage $\mathbf{z}$ sampled via the primitive vector $\mathbf{g}$. To this end, the perturbation generation function PERTURB is first called to produce a perturbation vector $\mathbf{p}$, which ensures spherical Gaussian distribution for the solution $\mathbf{y}$. To summarize, we have $\mathbf{A}\mathbf{y} = \beta$, where $\mathbf{y} \leftarrow D_{\Lambda_q(\mathbf{A}),\sigma_s}$, $\mathbf{p} \in \mathcal{R}^m$, and $\mathbf{z} \in \mathcal{R}^{\bar{m}}$, where $m = \bar{m} + 2$. For the implementation details of PERTURB and SAMPLEG functions with basis $b \geq 2$, see reference [18].

Algorithm 2 Gaussian preimage sampling [42]

```
function GAUSSSAMP($\mathbf{A}, (\boldsymbol{\rho}, \boldsymbol{\upsilon}), \beta, \sigma, \sigma_s$)
    $\mathbf{p} \leftarrow$ PERTURB($n, q, \sigma_s, (b+1)\sigma, (\boldsymbol{\rho}, \boldsymbol{\upsilon})$)
    $\mathbf{z} \leftarrow$ SAMPLEG($\sigma, \beta - \mathbf{A}\mathbf{p}, q$)
    $\mathbf{y} \leftarrow [p_1 + \boldsymbol{\upsilon}\mathbf{z}, p_2 + \boldsymbol{\rho}\mathbf{z}, p_3 + z_1, \ldots, p_m + z_{\bar{m}}]$
    return $\mathbf{y}$
end function
```

The parameter σ_s in PERTURB operation is referred as the *spectral norm*, which may be interpreted as a distribution parameter for Gaussian samples $\mathbf{y}$. The spectral norm in our implementation increases with base b. For the spectral norm parameter σ_s in the same algorithm, we use [8], [42]:

$$\sigma_s > s_1(\mathbf{X})\,\alpha,$$

where $\mathbf{X}$ is a subgaussian random matrix with parameter σ and $\alpha = (b+1)\sigma$.

Lemma 2.9 of [42] states that

$$s_1(\mathbf{X}) \leq C_0 \cdot \sigma \cdot \left(\sqrt{n\bar{m}} + \sqrt{2n} + t\right),$$

where C_0 is a constant and t is at most 4.7. We can now rewrite σ_s as

$$\sigma_s > C_0 \cdot (b+1) \cdot \sigma^2 \cdot \left(\sqrt{n\bar{m}} + \sqrt{2n} + 4.7\right), \tag{1}$$

where C_0 can be found empirically as the minimum value for which the perturbation covariance matrix is well-defined. In our experiments, this empirical value was selected as $C_0 = 1.3$.

In summary, using a larger base has an adverse affect on the performance of our lattice trapdoors by increasing the norm of the solution to $\mathbf{A}\mathbf{y} = \beta$. On the other hand, it can also improve the cryptographic schemes based on lattice trapdoors by enabling the use of much shorter trapdoors. Therefore, using higher bases not only improves the execution times of cryptographic algorithms in the subsequent sections, but also their storage requirements. In summary, its advantage generally outweighs its drawbacks; but the choice of the largest usable base depends on the cryptographic scheme for which the lattice trapdoor is used. In particular, the correctness and security constraints of the underlying cryptographic algorithms determine the largest base that can be used, as explained in Section IV-A.

IV. Cryptographic Schemes

In this section, we explain the ring constructions of three cryptographic applications: GVP signature, IBE and CP-ABE.

A. GPV Signature

The concept of GPV signature is first proposed in [27] and its ring-LWE version is described and implemented in [8]. Later, a more efficient implementation of GPV signature is presented in [32].

The GPV signature scheme consists of three functions: *Key Generation*, *Sign* and *Verify*. In key generation, user calls the trapdoor generation function (Algorithm 1) and obtains a public and secret key pair `(pk, sk)` $\leftarrow$ TRAPGEN(λ), where `pk` $= \mathbf{A}$ and `sk` $= \mathbf{T}_{\mathbf{A}}$.

The secret key is used to sign the hash h of a message μ, where $h \leftarrow H_{\texttt{sign}}(\mu)$ and $H_{\texttt{sign}} : \{0,1\}^* \rightarrow \mathcal{R}_q$. Then, the signature generation operation simply calls the Gaussian sampling function and obtains a short vector $\mathbf{x} \in \mathcal{R}^{(\bar{m}+2)}$, where $\mathbf{Ax} = h$: $\mathbf{x} \leftarrow \textsc{GaussSamp}(\mathbf{A}, \mathbf{T_A}, h, \sigma, s)$. The verification operation checks if $\mathbf{Ax} = H_{\texttt{sign}}(\mu)$ and $|\mathbf{x}| < \nu$, where ν is the norm bound for the signature.

Using higher base values increases the norm of the signature as can be observed in Eq 1. Apparently, the signature norm must be substantially smaller than the modulus q used in GPV signatures due to the security constraint imposed by the SIS problem (see Definition 2.3). To this end, Micciancio et al. [44] provide the following formula

$$\nu = 4^{\sqrt{n \log q \log \delta}} \tag{2}$$

to find the Euclidean norm ν of the signature given the root Hermite factor δ, which determines the security level. Using δ in the first row of Table I, we can conclude that the largest base is 8 for parameter $n = 512$ and $q \approx 2^{24}$ to maintain the same security level provided by $\delta = 1.006546$. Eq 1 can be used to compute the infinity norm of a signature. For instance, the infinity norm of a signature for $b = 8$, $n = 512$ and $q = 2^{24}$ is $\sigma_s \approx 15.06$; a 15-bit number. An upper bound for the Euclidean norm of the signature then can be computed using $\nu = \sqrt{n \cdot m} \cdot \sigma_s \approx 2188264$. Substituting ν in Eq. 2 results in $\delta = 1.006275$, which is smaller than the value in the first row of Table I. Any larger base results in a larger δ, which means a lower security level.

The second row in Table I represents a higher security level for GPV signatures with $n = 1024$ and $q \approx 2^{27}$. If we want to maintain the security level by $\delta = 1.003941$, the largest base is $b = 64$ resulting in $\delta = 1.00372$. However, even for $b = 512$, we have $\delta = 1.00484$, which provides substantially higher security level than GPV signature scheme with $n = 512$ and $q \approx 2^{24}$.

We also applied the security analysis provided in [53] and found out that the security levels for our choice of parameters are exactly the same as those obtained with the analysis in [44]. Consequently, in our implementation we use $b = 8$ and $b = 512$ for $(n, k) = (512, 24)$ and $(n, k) = (1024, 27)$, respectively. Note that the same parameter sets are also used in both [8] and [32] and therefore we can provide a fair comparison.

B. Identity-Based Encryption Scheme

The four functions of our IBE scheme are explained in this section. The main difference of our variant w.r.t. the original construction [27] is the use of the RLWE assumptions (polynomial rings) vs. the LWE assumptions (matrices of integers), which significantly reduces the computational and storage complexity (as illustrated in Table 2 of [8] for the GPV signature). Moreover, our implementation of the scheme also benefits from the efficient trapdoor sampling optimized for the power-of-two cyclotomic rings [25].

1) Setup: IBE Setup operation is simply the generation of a trapdoor given a security parameter λ. We use the TrapGen function in Algorithm 1 and master public and secret keys are set as follows

$$(\texttt{MPK},\ \texttt{MSK}) = (\mathbf{A}, \mathbf{T_A}) \leftarrow \textsc{TrapGen}(\lambda). \tag{3}$$

The private key generator (PKG) executes the TrapGen function, publishes the master public key MPK and keeps the master secret key MSK private as the latter is used to generate private keys of users.

2) Key Generation: In IBE scheme, the public key of a user can be chosen as any string that uniquely identifies the user such as e-mail address, telephone number etc. As we work in ring $\mathcal{R}_q$, a hash function is used to transform the bit string ID to a ring element: $H_{\texttt{IBE}} : \{0,1\}^* \rightarrow \mathcal{R}_q$. Assuming $\texttt{ID} \in \{0,1\}^*$ is the public key of a user, PKG executes the IBE key generation operation described in Algorithm 3 to generate the corresponding user private key $\boldsymbol{\omega}_{\texttt{ID}}$. Note that $\mathbf{A}\boldsymbol{\omega}_{\texttt{ID}} = \beta_{\texttt{ID}}$, where $\boldsymbol{\omega}_{\texttt{ID}} \in \mathcal{R}_q^{(\bar{m}+2)}$ is a short ring vector.

Algorithm 3 IBE Key Generation Algorithm

```
function IbeKeyGen($\mathbf{A}, \mathbf{T_A}, \texttt{ID}, \sigma, \sigma_s$)
    $\beta_{\texttt{ID}} \leftarrow H_{\texttt{IBE}}(\texttt{ID})$
    $\boldsymbol{\omega}_{\texttt{ID}} \leftarrow$ GaussSamp($\mathbf{A}, \mathbf{T_A}, \beta_{\texttt{ID}}, \sigma, \sigma_s$)
    return $\boldsymbol{\omega}_{\texttt{ID}}$
end function
```

Algorithm 4 IBE Encryption Algorithm

```
function IbeEnc($\mathbf{A}, \mathtt{ID}, \mu, \sigma$)
    $\beta_{\mathtt{ID}} \leftarrow H_{\mathtt{IBE}}(\mathtt{ID})$
    $s \leftarrow_U \mathcal{R}_q$
    $\mathbf{e}_0 \leftarrow D_{\mathcal{R}^m,\sigma}$
    $\mathbf{C}_0 \leftarrow \mathbf{A}^T s + \mathbf{e}_0$
    $e_1 \leftarrow D_{\mathcal{R},\sigma}$
    $c_1 \leftarrow \beta_{\mathtt{ID}} s + e_1 + \mu\lceil \frac{q}{2} \rceil$
    return $(\mathbf{C}_0, c_1)$
end function
```

3) Encryption: A message $\mu = (\mu_0, \mu_1, \ldots, \mu_{n-1})$ is represented as a polynomial in $\mathcal{R}_2$, $\mu = \mu_0 + \mu_1 x + \ldots + \mu_{n-1} x^{n-1}$, where $\mu_i \in \{0, 1\}$. Then it is encrypted under the recipient's public key as described in Algorithm 4.

From Algorithm 4, one can easily observe that IBE encryption is an RLWE adaptation of the dual Regev encryption system introduced in [27].

4) Decryption: The ciphertext message $(\mathbf{C}_0, c_1)$ encrypted under the public key $\mathtt{ID}$ can be decrypted using the corresponding private key $\boldsymbol{\omega}_{\mathtt{ID}}$ as described in Algorithm 5.

Algorithm 5 IBE Decryption Algorithm

```
function IbeDec($(\mathbf{C}_0, c_1), \boldsymbol{\omega}_{\mathtt{ID}}, q$)
    $t = c_1 - \boldsymbol{\omega}_{\mathtt{ID}}^T \cdot \mathbf{C_0}$
    for $i = 0$ to $n-1$ do
        if $\mid t_i \mid < \frac{q}{4}$ then $\bar{\mu}_i = 0$
        else $\bar{\mu}_i = 1$
        end if
    end for
    return $\bar{\mu}$
end function
```

5) Correctness: The correctness of the decryption algorithm can be easily verified as follows

$$\begin{aligned} c_1 - \boldsymbol{\omega}_{\mathtt{ID}}^T \cdot \mathbf{C_0} &= \beta_{\mathtt{ID}} s + e_1 + \mu\lceil \frac{q}{2} \rceil - (A\boldsymbol{\omega}_{\mathtt{ID}})^T s - \boldsymbol{\omega}_{\mathtt{ID}}^T \mathbf{e}_0 \\ &= \beta_{\mathtt{ID}} s + e_1 + \mu\lceil \frac{q}{2} \rceil - \beta_{\mathtt{ID}} s - \boldsymbol{\omega}_{\mathtt{ID}}^T \mathbf{e}_0 \\ &= e_1 + \mu\lceil \frac{q}{2} \rceil - \boldsymbol{\omega}_{\mathtt{ID}}^T \mathbf{e}_0. \end{aligned}$$

Provided that the norm of $\boldsymbol{\omega}_{\mathtt{ID}}^T \mathbf{e}_0$ is sufficiently small, the decryption process succeeds (i.e., $\mu = \bar{\mu}$). This is only possible if the private key $\boldsymbol{\omega}_{\mathtt{ID}}$ is a small norm ring vector.

In fact, we can estimate an upper bound for the norm of the polynomial $\boldsymbol{\omega}_{\mathtt{ID}}^T \mathbf{e}_0 \in \mathcal{R}_q$ if we know the upper bound for the private key $\boldsymbol{\omega}_{\mathtt{ID}}$, which is obtained via Gaussian preimage sampling function GaussSamp in Algorithm 2. Therefore, $\boldsymbol{\omega}_{\mathtt{ID}}$ follows a zero-centered Gaussian distribution with a standard deviation. Consequently, it is possible to provide an upper bound for the polynomial coefficients in the ring vector $\boldsymbol{\omega}_{\mathtt{ID}}$. As the noise $\mathbf{e}_0$ used in the encryption operation is also Gaussian, we can also find an upper bound for its norm. Suppose that Δ_ω and Δ_e are upper bounds for $\boldsymbol{\omega}_{\mathtt{ID}}$ and $\mathbf{e}_0$, respectively. Then a practical upper bound for $\boldsymbol{\omega}_{\mathtt{ID}} e_0$ (ignoring the factor e_1 as it is comparably negligible) can be estimated as $\Delta = \Delta_e \Delta_{\boldsymbol{\omega}} \sqrt{nm}$ utilizing the central limit theorem.

The error function $\mathtt{erf}\left(\frac{\Delta}{\sigma\sqrt{2}}\right)$ approximates the probability that a random sample from a zero-centered Gaussian distribution with distribution parameter σ lies between $-\Delta$ and Δ. Then, $1 - \mathtt{erf}\left(\frac{\Delta}{\sigma\sqrt{2}}\right)$ is the probability that the sample exceeds the upper bound Δ. For $\Delta = 8\sigma$, this probability is approximately $2^{-49.51}$. Therefore, we can use $\Delta_e = 8\sigma$ and $\Delta_\omega = 8\sigma_s$ as upper bounds for the norms of $\boldsymbol{\omega}_{\mathtt{ID}}$ and $\mathbf{e}_0$, respectively., Consequently, we can obtain the correctness constraint as

$$q > 256\sigma\sigma_s\sqrt{nm}. \tag{4}$$

The size of the modulus and the ring dimension are determined by both security and correctness constraints. Using Eq. 4 for $n = 1024$ we find out that the smallest bit size for q is 32 for base $b = 2$, while it is 39 bits for $b = 1024$. As can be observed in Table I, the lowest security level is more than 90 bits considering also a quantum computer attack. The correctness constraints are confirmed by the experimental results in Section V.

6) Security: We can prove the security of our IBE scheme variant using the approach similar to the one for the original matrix variant [27]. We first prove the semantic security under the random oracle model, followed by our scheme's security under RLWE assumptions.

Let $\mathcal{A}$ be an adversary attacking the IBE scheme with an advantage ϵ with the capability of querying H_{IBE} Q distinct times. Now, we construct an additional adversary, $\mathcal{S}$, and show that it has an advantage negligibly close to ϵ/Q, hence it is arguably the same as for $\mathcal{A}$. Let $\mathcal{S}$ try to simulate $\mathcal{A}$ as follows:

- $\mathcal{S}$ takes an input $\mathbf{A} \in \mathcal{R}_q^m$ and public key $\beta^*_{\mathtt{ID}} \in \mathcal{R}_q$, and generates an index $i \leftarrow_U [Q]$, where $\mathbf{A}$ is shared with $\mathcal{A}$.
- When $\mathcal{A}$ distinctly queries H_{IBE} for ID_j for the jth time, store the tuple $(ID_j, \beta^*_{\mathtt{ID}}, \perp)$ and return $\beta^*_{\mathtt{ID}}$ to $\mathcal{A}$ if $i = j$. If not; generate a public/secret key pair $(\beta_{\mathtt{ID}_j}, \omega_{\mathtt{ID}_j})$, where $\omega_{\mathtt{ID}_j} \leftarrow$ IBEKEYGEN, store the tuple $(ID_j, \beta_{\mathtt{ID}_j}, \omega_{\mathtt{ID}_j})$ and return $\beta_{\mathtt{ID}_j}$ to $\mathcal{A}$.
- When $\mathcal{A}$ asks for a secret key for ID, $\mathcal{S}$ assumes $\mathcal{A}$ queried H_{IBE} for ID. Then $\mathcal{S}$ retrieves the tuple $(ID, \beta_{\mathtt{ID}}, \omega_{\mathtt{ID}})$. $\mathcal{S}$ returns a random polynomial and aborts if $\omega_{\mathtt{ID}} = \perp$, returns $\omega_{\mathtt{ID}}$ if not.
- When $\mathcal{A}$ creates a challenge identity ID^* and messages μ_0 and μ_1, $\mathcal{S}$ assumes $\mathcal{A}$ queried H_{IBE} for ID^*. If the tuple $(ID^*, \beta^*_{\mathtt{ID}}, \perp)$ is not stored, $\mathcal{S}$ returns a random polynomial and aborts. If it is, send the messages μ_0 and μ_1 to the challenger, receives the challenge ciphertext c_1^* and sends it to $\mathcal{A}$.

Assuming i is hidden from $\mathcal{A}$, the probability that $\mathcal{S}$ aborts during the simulation is $1/Q$. If we focus on the cases that $\mathcal{S}$ does not abort, we can claim that the interaction it has with $\mathcal{A}$ is statistically close to an interaction of $\mathcal{A}$ with the real IBE system, due to the uniform nature of $\beta_{\mathtt{ID}}$ and $\omega_{\mathtt{ID}}$ generated by $\mathcal{S}$. We can then observe that for the cases that $\mathcal{S}$ does not abort, it has the same advantage as $\mathcal{A}$.

Now, knowing that the random oracle model is secure, we can proceed to show that our IBE scheme is IND-CPA-secure under RLWE assumptions. We start by showing the secret s cannot be recovered. Let $\mathcal{A}$ be an adversary who has access to $\mathbf{A}$, H_{IBE}, the ciphertext pair $(C_0, c_1))$, and the encryption oracle. Recall that $\mathbf{C}_0 = \mathbf{A}^T s + \mathbf{e}_0 \in \mathcal{R}_q^m$ and $c_1 = \beta_{\mathtt{ID}} s + e_1 + \mu\lceil \frac{q}{2} \rceil \in \mathcal{R}_q$. Adversary $\mathcal{A}$ can try to recover s from both $\mathbf{C}_0$ and c_1. Recall that $\mathbf{A}$ is uniformly random, which means $\mathbf{A}^T$ also is. We can represent $\mathbf{A}^T$ as $\mathbf{A}^T = [a_0, a_1, ..., a_{m-1}]$ and e_0 as $e_0 = [e_{0_0}, e_{0_1}, ..., e_{0_{m-1}}]$, which means $\mathbf{C}_0 = [a_0 s + e_{0_0}, a_1 s + e_{0_1}, ..., a_{m-1} s + e_{0_{m-1}}]$. As $\mathcal{A}$ has access to both $\mathbf{A}$ and $\mathbf{C}_0$, it can generate tuples in the form $(a_i s + e_{0_i}, a_i)$, where $0 \leq i \leq m-1$. Since $\mathbf{A}^T$ is uniformly random and $\mathbf{e}_0 \leftarrow D_{\mathcal{R}^m,\sigma}$, solving for tuples $(a_i s + e_{0_i}, a_i)$ is as hard as solving the RLWE problem (for the Search Assumption 2.4), hence $\mathcal{A}$ cannot recover s from $\mathbf{C}_0$. Similarly, having access to c_1 and H_{IBE}, $\mathcal{A}$ can generate tuples $(\beta_{\mathtt{ID}} s + e_1 + \mu\lceil \frac{q}{2} \rceil, \beta_{\mathtt{ID}})$. Since H_{IBE} has uniformly random output by definition and $e_1 \leftarrow D_{\mathcal{R},\sigma}$, we can use the same method to show that $\mathcal{A}$ cannot recover s from c_1 based on the Search Assumption. If the adversary can solve $\omega_{\mathtt{ID}}$, it can also recover s. However, this would require solving $\mathbf{A}\omega_{\mathtt{ID}} = \beta_{\mathtt{ID}}$, which is equivalent to solving an instance of $\mathtt{SIVP}_\gamma$.

We can now show the indistinguishability of ciphertexts to complete our IND-CPA proof. Let μ_0 and μ_1 be arbitrary messages and $\mathcal{A}$ is challenged to output the correct $\kappa \in \{0, 1\}$ given $\mathbf{C}_0^* = \mathbf{A}^T s + \mathbf{e}_0$ and $c_1^* = \beta_{\mathtt{ID}} s + e_1 + \mu_\kappa \lceil \frac{q}{2} \rceil$, where κ is chosen uniformly randomly by the challenger. Since $\mathcal{A}$ cannot recover s, it cannot calculate the ciphertexts for μ_0 and μ_1 by itself. Using the same approach as above, we can think of $\mathbf{C}_0$ as $\mathbf{C}_0 = [a_0 s + e_{0_0}, a_1 s + e_{0_1}, ..., a_{m-1} s + e_{0_{m-1}}]$. By the Decision Assumption of RLWE (see Definition 2.5), $\mathcal{A}$ cannot distinguish between $(a_i s + e_{0_i})$ and b_i, where $b_i \leftarrow_U \mathcal{R}_q$, which means it cannot also distinguish between $\mathbf{C}_0$ and $\mathbf{B}_i$, where $\mathbf{B}_i \leftarrow_U \mathcal{R}_q^m$. Similarly, by the same assumption $\mathcal{A}$ cannot distinguish between c_1 and b_i. Let $(\mathbf{C}_{0_\kappa}, c_{1_\kappa})$ be the ciphertext pair generated from the message μ_κ. By the Decision Assumption $\mathcal{A}$ cannot distinguish between $(\mathbf{C}_{0_0}, c_{1_0})$ and $(\mathbf{B}_i, b_i)$ as well as $(\mathbf{C}_{0_1}, c_{1_1})$ and $(\mathbf{B}_i, b_i)$, which means that it cannot distinguish between $(\mathbf{C}_{0_0}, c_{1_0})$ and $(\mathbf{C}_{0_1}, c_{1_1})$. Thus $\mathcal{A}$ cannot succeed in determining κ with any non-negligible advantage.

C. Ciphertext-Policy Attribute-Based Encryption Scheme

In this section we provide the details for our CP-ABE scheme. The main difference of our variant w.r.t. the original construction [58] is the use of the RLWE assumptions (polynomial rings) vs. the LWE assumptions (matrices of integers), which significantly reduces the computational and storage complexity (as illustrated in Table 2 of [8] for the GPV signature). Moreover, our implementation of the scheme also benefits from the efficient trapdoor sampling optimized for the power-of-two cyclotomic rings [25].

The scheme supports access policies that can be expressed as conjunctions over a subset of positive and negative attributes. A positive attribute in an access policy requires that user have that attribute to decrypt a ciphertext encrypted under that policy. Negative attributes, on the other hand, are used to exclude a certain set of users from decrypting the ciphertext generated under that access policy. We use symbols $+$ and $-$ in superscript to denote positive and negative attributes, respectively.

The essential idea in CP-ABE is that PKG generates a secret key for each user in the system based on user's attributes. For this, PKG first generates a public key $\mathbf{A}$ and a corresponding trapdoor $\mathbf{T_A}$ in the setup function.

1) Setup: PKG uses Algorithm 6 to generate master public and master secret keys: `MPK` and `MSK`. After generating

Algorithm 6 CP-ABE Setup Algorithm

```
function CpeABESetup($\ell, \lambda$)
    $(\mathbf{A}, \mathbf{T_A}) \leftarrow$ TrapGen($\lambda$)
    $\beta \leftarrow_U R_q$
    for $i = 1$ to $\ell$ do
        $(\mathbf{B}_i^+, \mathbf{B}_i^-) \leftarrow_U R_q^{1\times m}$
    end for
    MPK $\leftarrow \{\mathbf{A}, \{\mathbf{B}_i^+, \mathbf{B}_i^-\}_{i\in[\ell]}, \beta\}$
    MSK $\leftarrow \mathbf{T_A}$
    return (MPK, MSK)
end function
```

the public vector $\mathbf{A} \in \mathcal{R}_q^{1\times m}$, the corresponding trapdoor $\mathbf{T_A}$ and uniformly generated public challenge β, PKG generates a uniformly distributed pair of vectors $(\mathbf{B}_i^+, \mathbf{B}_i^-)$ for each attribute in the universal set of attributes $\mathcal{X} = \{x_1, x_2, \ldots, x_\ell\}$, where $\mathbf{B}_i^+, \mathbf{B}_i^- \in \mathcal{R}_q^{1\times m}$ for $i = 1, \ldots, \ell$. Alternatively, we can employ a hash function $H_{CP-ABE} : \mathcal{X} \to \mathcal{R}_q^{2\times m}$ for each attribute: $(\mathbf{B}_i^+, \mathbf{B}_i^-) \leftarrow H_{CP-ABE}(x_i)$ for $i = 1, \ldots, \ell$.

2) Key Generation: PKG generates the private key of a user holding an attribute set $\mathcal{Y} \subseteq \mathcal{X}$ as depicted in Algorithm 7. Private key components $\boldsymbol{\omega}_i \in \mathcal{R}^m$ for $i = 1, \ldots, \ell$, corresponding to attributes in $\mathcal{X}$ are sampled directly from a discrete Gaussian distribution. Then, depending on the attribute subset held by the user, a new challenge η is calculated. PKG is the only party in the system that can generate a short solution to $\mathbf{A}\boldsymbol{\omega}_{\mathbf{A}} = \eta$ as it knows the trapdoor. It is easy to see that

$$(\mathbf{A}, \tilde{\mathbf{B}}_{\mathbf{1}}, \ldots, \tilde{\mathbf{B}}_\ell)\boldsymbol{\omega}_{\mathcal{Y}}^T = \beta, \tag{5}$$

where $\tilde{\mathbf{B}}_{\mathbf{i}} = \mathbf{B}_{\mathbf{i}}^+$ if $i \in \mathcal{Y}$, otherwise $\tilde{\mathbf{B}}_{\mathbf{i}} = \mathbf{B}_{\mathbf{i}}^-$

3) Encryption: A sender determines an access policy $\mathcal{W} = (\mathcal{W}^+ \cup \mathcal{W}^-)$, which can contain negative as well as positive attributes. The encryption algorithm depicted in Algorithm 8 takes the message $\mu \in \mathcal{R}_2$, the public key `MPK`, and the access policy $\mathcal{W}$ and outputs the ciphertext $\mathbf{C}$. The access policy is also output as a part of the ciphertext. Note that the length of the ciphertext depends on the access policy.

4) Decryption: The receiver uses Algorithm 9 to decrypt the ciphertext

$$\mathbf{C} = (\mathcal{W}, \mathbf{C}_{A,0}, \{\mathbf{C}_{0,i}\}_{i\in\mathcal{W}}, \{\mathbf{C}_{0,i}^+, \mathbf{C}_{0,i}^-\}_{i\in\mathcal{X}\setminus\mathcal{W}}, c_1).$$

The decryption algorithm takes also the attribute set of the receiver $\mathcal{Y}$ and if $\mathcal{Y} \vdash \mathcal{W}$, decryption returns the original message μ, otherwise $\perp$. $\mathcal{Y} \vdash \mathcal{W}$ if $\mathcal{Y} \cap \mathcal{W}^+ = \mathcal{W}^+$ and $\mathcal{Y} \cap \mathcal{W}^- = \emptyset$.

Algorithm 7 CP-ABE Key Generation Algorithm

```
function CpeAbeKeyGen(MSK, MPK, $\ell, \mathcal{Y}, \sigma, \sigma_s$)
    $\omega = 0$
    for $i = 1$ to $\ell$ do
        $\boldsymbol{\omega}_i \leftarrow D_{\mathcal{R}_q^m, \sigma_s}$
        if $i \in \mathcal{Y}$ then $\eta \leftarrow \eta + \mathbf{B}_i^+ \boldsymbol{\omega}_i$
        else $\eta \leftarrow \eta + \mathbf{B}_i^- \boldsymbol{\omega}_i$
        end if
    end for
    $\eta \leftarrow \beta - \eta$
    $\boldsymbol{\omega}_A \leftarrow$ GaussSamp($\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \eta, \sigma, \sigma_s$)
    $\boldsymbol{\omega}_{\mathcal{Y}} \leftarrow (\boldsymbol{\omega}_A, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_\ell)$
    return $\boldsymbol{\omega}_{\mathcal{Y}}$
end function
```

Algorithm 8 CP-ABE Encryption Algorithm

```
function CpeAbeEnc($\mu$, MPK, $\mathcal{W}, \sigma$)
    $s \leftarrow_U R_q$
    $e_1 \leftarrow D_{R,\sigma}$
    $c_1 \leftarrow s\beta + e_1 + \mu\lceil \frac{q}{2} \rceil$
    $\mathbf{e}_{0,A} \leftarrow D_{R^m,\sigma}$
    $\mathbf{C}_{0,A} \leftarrow \mathbf{A}^T s + \mathbf{e}_{0,A}$
    for $i = 1$ to $\ell$ do
        if $i \in \mathcal{W}^+$ then
            $\mathbf{e}_{0,i} \leftarrow D_{R^m,\sigma}$
            $\mathbf{C}_{0,i} \leftarrow (\mathbf{B}^+)^T s + \mathbf{e}_{0,i}$
        else if $i \in \mathcal{W}^-$ then
            $\mathbf{e}_{0,i} \leftarrow D_{R^m,\sigma}$
            $\mathbf{C}_{0,i} \leftarrow (\mathbf{B}^-)^T s + \mathbf{e}_{0,i}$
        else
            $\mathbf{e}_{0,i}^+, \mathbf{e}_{0,i}^- \leftarrow D_{R^m,\sigma}$
            $\mathbf{C}_{0,i}^+ \leftarrow (\mathbf{B}^+)^T s + \mathbf{e}_{0,i}^+$
            $\mathbf{C}_{0,i}^- \leftarrow (\mathbf{B}^-)^T s + \mathbf{e}_{0,i}^-$
        end if
    end for
    $\mathbf{C} \leftarrow (\mathcal{W}, \mathbf{C}_{A,0}, \{\mathbf{C}_{0,i}\}_{i\in\mathcal{W}}, \{\mathbf{C}_{0,i}^+, \mathbf{C}_{0,i}^-\}_{i\in\mathcal{X}\setminus\mathcal{W}}, c_1)$
    return $\mathbf{C}$
end function
```

5) Correctness: If $\mathcal{Y} \cap \mathcal{W}^+ = \mathcal{W}^+$ and $\mathcal{Y} \cap \mathcal{W}^- = \emptyset$ then we have the following equations

$$\begin{aligned} a =& \boldsymbol{\omega}_A(\mathbf{A}^T s) + \boldsymbol{\omega}_A \mathbf{e}_{0,A} + \boldsymbol{\omega}_1(\tilde{\mathbf{B}}_1^T s) + \boldsymbol{\omega}_1 \mathbf{e}_{0,1} + \\ & \ldots + \boldsymbol{\omega}_\ell(\tilde{\mathbf{B}}_\ell^T s) + \boldsymbol{\omega}_\ell \mathbf{e}_{0,\ell} \\ =& ((\mathbf{A}\boldsymbol{\omega}_A)^T s) + \boldsymbol{\omega}_A \mathbf{e}_{0,A} + ((\tilde{\mathbf{B}}_1 \boldsymbol{\omega}_1)^T s) + \boldsymbol{\omega}_1 \mathbf{e}_{0,1} + \\ & \ldots + ((\tilde{\mathbf{B}}_\ell \boldsymbol{\omega}_\ell)^T s) + \boldsymbol{\omega}_\ell \mathbf{e}_{0,\ell} \\ =& \beta s + \boldsymbol{\omega}_A \mathbf{e}_{0,A} + \boldsymbol{\omega}_1 \mathbf{e}_{0,1} + \ldots + \boldsymbol{\omega}_\ell \mathbf{e}_{0,\ell}, \end{aligned}$$

where $\tilde{\mathbf{B}}_i \in \{\mathbf{B}_i^+, \mathbf{B}_i^-\}$ Consequently, we have

$$c_1 - a = e_1 + \mu\lceil \frac{q}{2} \rceil - \boldsymbol{\omega}_A \mathbf{e}_{0,A} - \boldsymbol{\omega}_1 \mathbf{e}_{0,1} - \ldots - \boldsymbol{\omega}_\ell \mathbf{e}_{0,\ell}. \tag{6}$$

Algorithm 9 CP-ABE Decryption Algorithm

```
function CpeAbeDec($\mathbf{C}$, MPK, $\mathcal{Y}$)
    $a \leftarrow (\mathbf{C}_A)^T \boldsymbol{\omega}_A$
    for $i = 1$ to $\ell$ do
        if $i \in W$ then $a \leftarrow a + (\mathbf{C}_{0,i})^T \boldsymbol{\omega}_i$
        else
            if $i \in \mathcal{Y}$ then $a \leftarrow a + (\mathbf{C}^{+}_{0,i})^T \boldsymbol{\omega}_i$
            else $a \leftarrow a + (\mathbf{C}^{-}_{0,i})^T \boldsymbol{\omega}_i$
            end if
        end if
    end for
    $t \leftarrow c_1 - a$
    for $i = 0$ to $n$ do
        if $\mid t_i \mid < \frac{q}{4}$ then $\mu_i \leftarrow 0$
        else $\mu_i \leftarrow 1$
        end if
    end for
    return $\mu$
end function
```

TABLE II
MODULUS BIT SIZES $k = \lceil \log_2 q \rceil$ FOR DIFFERENT NUMBERS OF ATTRIBUTES AND BASES

base b	k $\ell =$ (6 / 8 / 16 / 20 / 32)
2	34 / 34 / 35 / 35 / 35
4	34 / 34 / 35 / 35 / 35
8	34 / 35 / 35 / 35 / 36
16	35 / 35 / 36 / 36 / 36
32	36 / 36 / 37 / 37 / 37
64	37 / 37 / 37 / 38 / 38
128	38 / 38 / 38 / 38 / 39
256	38 / 39 / 39 / 39 / 40
512	39 / 40 / 40 / 40 / 41
1024	40 / 40 / 41 / 41 / 42

In Eq. 6, all private key components, $\boldsymbol{\omega}_i$ for $i = 0, \ldots, \ell$ (except for $\boldsymbol{\omega}_A$) are directly sampled from the same distribution as $\boldsymbol{\omega}_A$. Therefore, an upper bound for each $\boldsymbol{\omega}_i$ can be taken as the same upper bound for $\boldsymbol{\omega}_A$.

On the other hand, the private key $\boldsymbol{\omega}_A$ is generated using the Gaussian sampler in Algorithm 2. We can formulate an upper bound for all noise factors combined in Eq. 6 (ignoring e_1) as follows $\Delta = \Delta_e \Delta_\omega \sqrt{nm(\ell+1)}$, If $\Delta < \frac{q}{4}$, then the decryption is possible. Therefore, the correctness constraint can be written as

$$q > 256\sigma\sigma_s\sqrt{nm(\ell+1)}) \tag{7}$$

Eq. 7 suggests that the correctness constraint is affected by the number of attributes and therefore we have to increase the modulus size with the number attributes resulting in a lower security level. Table II lists the required modulus sizes for different values of the base and the number of attributes.

6) Security: We can prove that the CP-ABE scheme is secure against selective chosen plaintext attack (sCPA) by adapting the security game in [58] to our RLWE construction. Before the security proof, we recall that the pair $(a_i, a_i s + e_i)$ is pseudorandom (i.e., indistinguishable from a uniformly random pair based on the hardness of the decision RLWE problem) for an arbitrary $s \in \mathcal{R}_q$, uniformly random $a_i \leftarrow_U \mathcal{R}_q$ and $e_i \leftarrow D_{\mathcal{R},\sigma}$ and $i = 1, \ldots, t$.

We can sketch a simple security game, in which an RLWE solver $\mathcal{B}$ has an oracle $\mathcal{O}$. In the game, either pseudorandom or uniformly random ring elements (or vectors) are selected and $\mathcal{B}$ is challenged to tell the distribution. Suppose there exists a polynomial adversary $\mathcal{A}$ that breaks sCPA security of the CP-ABE scheme with an advantage ϵ. Then, we can show that $\mathcal{B}$ solves RLWE problem, which is the decision RLWE problem in this context.

For this, $\mathcal{B}$ is challenged with an access policy $\mathcal{W}^* = \mathcal{W}^+ \cup \mathcal{W}^-$ in the security game. $\mathcal{B}$ on the other hand should be able to simulate the view of $\mathcal{A}$ for other access policies except for $\mathcal{W}^*$. The security game proceeds as in the following steps.

- *Commitment Phase:* Adversary $\mathcal{A}$ commits to an access policy $\mathcal{W}^* = \mathcal{W}^+ \cup \mathcal{W}^-$ and sends it to $\mathcal{B}$.
- *Setup Phase:* $\mathcal{B}$ and $\mathcal{O}$ interacts as described here. The key point here is that $\mathcal{O}$ uses either pseudorandom or uniformly random distributions to respond to the queries of $\mathcal{B}$.
  - $\mathcal{B}$ obtains $(\mathbf{A}, \mathbf{V}_\mathbf{A}) \in \mathcal{R}_q^{1\times m} \times \mathcal{R}_q^m$ and $(u, v_u) \in \mathcal{R}_q \times \mathcal{R}_q$ from $\mathcal{O}$.
  - For each $i \in \mathcal{X}^\ell \setminus \mathcal{W}^*$, $\mathcal{B}$ obtains $(\mathbf{B}_i^+, \mathbf{V}_i^+), (\mathbf{B}_i^-, \mathbf{V}_i^-) \in \mathcal{R}_q^{1\times m} \times \mathcal{R}_q^m$ from $\mathcal{O}$.
  - For each $i \in \mathcal{W}^+$, $\mathcal{B}$ obtains $(\mathbf{B}_i^+, \mathbf{V}_i^+) \in \mathcal{R}_q^{1\times m} \times \mathcal{R}_q^m$ from $\mathcal{O}$; but computes $(\mathbf{B}_i^-, \mathbf{T}_{B_i}^-) \leftarrow \textsc{TrapGen}(\lambda)$.
  - For each $i \in \mathcal{W}^-$, $\mathcal{B}$ obtains $(\mathbf{B}_i^-, \mathbf{V}_i^-) \in \mathcal{R}_q^{1\times m} \times \mathcal{R}_q^m$ from $\mathcal{O}$; but computes $(\mathbf{B}_i^+, \mathbf{T}_{B_i}^+) \leftarrow \textsc{TrapGen}(\lambda)$.
  - $\mathcal{B}$ publishes $\mathtt{MPK} = \{\mathbf{A}, \{\mathbf{B}_i^+, \mathbf{B}_i^-\}_{i\in[\ell]}, u\}$; keeps $(\{\mathbf{T}_{B_i}^-, \mathbf{V}_i^+\}_{i\in\mathcal{W}^+}, \{\mathbf{T}_{B_i}^+, \mathbf{V}_i^-\}_{i\in\mathcal{W}^-},$ $\{\mathbf{V}_i^+, \mathbf{V}_i^-\}_{i\in\mathcal{X}^\ell\setminus\mathcal{W}^*})$ secret.
- *Key Generation Queries:* In this phase, $\mathcal{B}$ simulates the view of $\mathcal{A}$ by responding $\mathcal{A}$'s key generation queries for an access policy $\mathcal{W} \neq \mathcal{W}^*$. For $\mathcal{W} \neq \mathcal{W}^*$, we have $\mathcal{W} \cap \mathcal{W}^+ \neq \mathcal{W}^+$ or $\mathcal{W} \cap \mathcal{W}^= \neq \emptyset$. This implies that $\mathcal{B}$ knows at least one $\mathbf{T}_{B_i^+}$ or $\mathbf{T}_{B_i^-}$. Then it can compute $\boldsymbol{\omega}_\mathcal{Y}$ for any attribute subset $\mathcal{Y} \vdash \mathcal{W}$. $\mathcal{A}$ can make more than one query.
- *Challenge:* $\mathcal{A}$ picks two random messages $(\mu_0, \mu_1) \in \mathcal{R}_2$ and sends them to $\mathcal{B}$, which selects one of them at random and computes $c_1 = v_u + \mu_\kappa \lceil \frac{q}{2} \rceil$ with $\kappa \in \{0, 1\}$. It also sets $\mathbf{C}_{0,A} = \mathbf{V}_\mathbf{A}$; $\mathbf{C}_{0,i} = \mathbf{V}_i^+$ for each $i \in \mathcal{W}^+$; $\mathbf{C}_{0,i} = \mathbf{V}_i^-$ for each $i \in \mathcal{W}^-$. Then for each $i \in \mathcal{X}^\ell \setminus \mathcal{W}^*$, it sets $\mathbf{C}_{0,i}^+ = \mathbf{V}_i^+$ and $\mathbf{C}_{0,i}^- = \mathbf{V}_i^-$. $\mathcal{B}$ returns $\mathbf{C}^* = (\mathcal{W}^*, \mathbf{C}_{0,A}, \{\mathbf{C}_{0,i}\}_{i\in\mathcal{W}^*}, \{\mathbf{C}_{0,i}^+, \mathbf{C}_{0,i}^-\}_{i\in\mathcal{X}\setminus\mathcal{W}^*}, c_1)$ to $\mathcal{A}$.

At the end of the security game, adversary $\mathcal{A}$ outputs κ. If $\mathcal{O}$ is a pseudorandom oracle, $\mathbf{C}^*$ is a valid ciphertext and therefore $\mathcal{A}$ outputs the correct κ with an ϵ advantage. Otherwise, the ciphertext is uniformly random; therefore, $\mathcal{A}$ can only make a random guess and only succeed with a probability 1/2 (with no advantage). This means that $\mathcal{B}$ can distinguish whether $\mathcal{O}$ is a pseudorandom or uniformly random oracle, which breaks the decision RLWE hardness assumption (see Definition 2.5). Therefore, our assumption that $\mathcal{A}$ can break sCPA contradicts with the hardness assumption of the decision RLWE.

V. Implementation Details and Results

A. Software Implementation

We implemented the GPV signature, IBE, and CP-ABE schemes in the PALISADE library[2] [7], [12], [18], [32], [33], [48], which is a modular open-source lattice-based cryptography library. The library uses native data types, but does not employ any platform-specific optimizations, such as assembly-level routines.

PALISADE uses a layered approach with four software layers, each including a collection of C++ classes to provide encapsulation, low inter-class coupling and high intra-class cohesion. The software layers are as follows:

1) The cryptographic layer supports cryptographic protocols such as homomorphic encryption schemes through calls to lower layers.
2) The encoding layer supports plaintext encodings for cryptographic schemes.
3) The lattice constructs layer supports power-of-two and arbitrary cyclotomic rings (coefficient, Chinese Remainder Theorem (CRT), and double-CRT representations). Lattice operations are decomposed into primitive arithmetic operations on integers, vectors, and matrices here.
4) The arithmetic layer provides basic modular operations (multiple multiprecision and native math backends are supported), implementations of Number-Theoretic Transform (NTT), Fermat-Theoretic Transform (FTT), and Bluestein FFT. The integer distribution samplers are implemented in this layer.

[2]https://git.njit.edu/palisade/PALISADE

The work in this paper adds several new capabilities to a new PALISADE module called "trapdoor" (the module is expected to be publicly released with PALISADE v1.3 in the third quarter of 2018), which includes the following new features broken down by layer:

- Implementations of the GPV signature, IBE, and CP-ABE schemes in the cryptographic layer.
- Trapdoor sampling, including RLWE trapdoor generation, G-sampling and perturbation generation routines in the lattice layer. Cyclotomic fields $\mathcal{K}_{2n}$ and additional polynomial/double-CRT operations, such as polynomial transposition, are also in this layer.
- Generic integer Gaussian samplers and a Cooley-Tukey transform based on complex roots of unity in the arithmetic layer.

Several lattice-layer and arithmetic-layer optimizations are also applied for runtimes improvements.

Our implementation keeps $\mathcal{R}_q$ elements in the evaluation representation since the arithmetic over such representation is performed component-wise and therefore very fast. We always sample (using both uniformly random and integer Gaussian distribution) in polynomial representation and then convert the sample immediately to the evaluation representation. Therefore, any sampling operation in the algorithm requires one sampling followed by a NTT operation. Thereafter, the operands in cryptographic algorithms are usually kept in the evaluation representation until the decryption operation.

Integer Gaussian sampling is the primitive operation that is called repeatedly in many algorithms described in this paper. Thus, the selection and efficient implementation of the Gaussian sampling operation is of paramount importance for the overall performance of cryptographic operations; essentially signature generation in the GPV signature scheme, SETUP, KEY GENERATION and ENCRYPTION in both the IBE and the CP-ABE schemes.

An integer Gaussian generator returns a sample statistically close to $\mathcal{D}_{\mathbb{Z},c,\sigma}$. When the center c does not change and distribution parameter is relatively small, implementation of the inversion sampling method developed in [46] is a very good candidate as it is based on fast table lookups; i.e., the expensive floating-point exponentiation operation is never executed. On the other hand, when the center changes, the pre-computation technique employed in the inversion sampling cannot be used. For this we use the generic sampling method proposed by Karney [35], which proves to be a more efficient method when the distribution parameter is large and the center varies, as is the case for the trapdoor preimage sampling with large bases.

We implemented all the algorithms in standard C++ 11 with no architectural support such as optimized assembly language routines. We tested and evaluated them on a computer featuring an Intel(R) Core(TM) i7-3770 CPU with a 3.40 GHz clock frequency running Linux CentOS 7 operating system. We give the implementation results and comparisons for GPV signatures, IBE and CP-ABE in the subsequent sections. We use the single-thread mode to report execution times, which are calculated as the average of one hundred runs with randomly chosen inputs. We included storage requirements and execution timings for different bases. In all our implementations we used $\sigma = 4.57825$ as the distribution parameter for integer Gaussian sampling operations.

In the GPV signature scheme, the largest base that can be used is determined by the security constraint expressed in Eq (2). A large base increases the signature norm for a given set of (n, q), which decreases the security level. In IBE and CP-ABE schemes, using a higher base increases the norm of secret keys, which is the determining factor in the correctness constraints in Eqs (4) and (7). We use the highest base values for our implementation of the three schemes that achieve at least the minimum security level in Table I.

The number of attributes affects the performance of the encryption and decryption operations of CP-ABE. We use 32 as the maximum number of attributes in our CP-ABE experiments as no other work with a higher number of attributes has been reported in the literature.

Although this paper focuses on a software implementation of trapdoor sampling and GPV signature, IBE, and CP-ABE schemes, the algorithms implemented in our work can be utilized in compute environments with hardware accelerators, similar to the design proposed for an FPGA-accelerated homomorphic encryption co-processor by Cousins *et al.* [19]. The schemes discussed in our work are based on the same polynomial arithmetic operations for power-of-two cyclotomic rings, e.g., NTTs, ring additions, and ring multiplications. The polynomial arithmetic operations can also be offloaded to GPUs using the same approach as presented by Dai *et al.* [20] for key-policy ABE based on trapdoor sampling. The relatively small size of keys, ciphertexts, and precomputed parameters for all schemes presented in this paper suggests that the keys and ciphertexts can be stored/manipulated on embedded devices.

TABLE III
STORAGE REQUIREMENTS OF GPV SIGNATURE SCHEME FOR DIFFERENT BASES

Base	Public key & Signature	Private key
$n = 512, \lceil \log_2 q \rceil = 24$		
2	39 KB	72 KB
8	15 KB	24 KB
$n = 1024, \lceil \log_2 q \rceil = 27$		
2	116 KB	216 KB
64	28 KB	40 KB
512	20 KB	24 KB

B. Implementation Results for GPV Signature Scheme

In this section, we provide the execution times and storage requirements of the GPV signature scheme and show how using higher bases improves them.

Table III lists the storage requirements in bytes for public/private keys and the signature lengths for different bases and two security levels. In [8] and [32], where GPV signature implementations in software are reported, the storage requirements are the same as ours for base 2. Our implementation shortens public key and signature lengths by a factor of 2.6 and private key length by a factor of 3.0 for the case of $(512, 24)$. The factors of improvements for the case of $(1024, 27)$ are 5.8 and 9.0 for public key/signature and private key, respectively.

In Table IV, we give the execution times of the key generation, signature generation and verification operations in comparison with those in works [8], [24], [32]. In the case of $(512, 24)$, using the base of 8 (as compared to the binary base) improves the key generation, signature generation and verification runtimes by the factors of 3.78, 2.17, and 2.43, respectively. In the case of $(1024, 27)$, the improvements for the base of 512 are 7.92, 2.49, and 3.5, respectively, for the same operations. In all operations for both scenarios, the execution times of our implementation outperform those in the works [8] and [32], which are based on the same trapdoor sampling approach. Compared to [24], our key generation times are faster for the same ring dimensions while verification times are comparable. Our signing times, however, are slower, which can be expected considering the work [24] utilizes a different trapdoor sampling algorithm (based on NTRU lattices).

Furthermore, the signature generation operation can be partitioned into two phases: offline and online, where the offline phase does not depend on the message. Table IV provides both execution times for signature generation (first operand in the sum is the offline timing). Using the two-phase signature generation approach, our implementation performs one signature generation operation in as low as 2.16 ms whereas the best timings for GPV signature generation reported in the literature are 27 ms in both [32] and [8].

C. Implementation Results for IBE

In this section, we report storage requirements and timing results for our IBE implementation and compare them with those in [22] and its more recent implementation in [41], which are both based on lattices. We note that the comparison is not fair as the hardness assumptions and therefore trapdoor constructions are different. Our construction uses only classical standard RLWE assumptions whereas the construction in [22] (and its optimized version [41]) relies on a non-standard NTRU assumption along with standard RLWE assumptions. Therefore, there is a need for deeper analysis into the hardness assumptions of this non-standard NTRU problem.

Furthermore, our trapdoor construction is versatile in the sense that it can be used in other more advanced cryptographic applications, such as ABE, as shown in the next section (see also the key-policy attribute-based encryption in [11] that can be implemented using our trapdoor construction). On the other hand, there is no ABE scheme based on the construction in [22] or [41]. We provide the comparison, all the same, to give an idea as to how our construction compares with the state-of-the-art in the literature. We do not include a comparison with schemes based on classical hardness assumptions such as those in bilinear pairings, which are not post-quantum. Such a comparison is available in [22] showing that the lattice-based IBE is comparable to pairing-based IBE schemes from the execution time perspective, while it does not fare well in terms of storage requirements.

TABLE IV
SINGLE-THREADED RUNTIMES (IN MS) FOR GPV SIGNATURE AT DIFFERENT BASES AND COMPARISON WITH PRIOR RESULTS

Base	**Key Gen.**	**Sign**	**Verification**
this work $n = 512$, $\lceil \log_2 q \rceil = 24$ @3.4 GHz			
2	3.63	$9.69 + 8.60 = 18.29$	0.17
8	0.96	$5.91 + 2.51 = 8.42$	0.07
[8] $n = 512$, $\lceil \log_2 q \rceil = 24$ @2.3 GHz			
2	4,562	27	3.00
[32] $n = 512$, $\lceil \log_2 q \rceil = 24$ @3.4 GHz			
2	9.5	27	0.33
[24] $n = 512$, $\lceil \log_2 q \rceil = 14$ @3.3 GHz			
NA	6.98	0.16	0.03
this work $n = 1024$, $\lceil \log_2 q \rceil = 27$ @3.4 GHz			
2	5.86	$16.92 + 15.50 = 32.42$	0.28
64	1.23	$11.81 + 3.31 = 15.12$	0.10
512	0.74	$10.86 + 2.16 = 13.02$	0.08
[8] $n = 1024$, $\lceil \log_2 q \rceil = 27$ @2.3 GHz			
2	28,074	74	10.00
[32] $n = 1024$, $\lceil \log_2 q \rceil = 27$ @3.4 GHz			
2	17.2	62.5	0.68
[24] $n = 1024$, $\lceil \log_2 q \rceil = 14$ @3.3 GHz			
NA	19.64	0.33	0.06

TABLE V
STORAGE REQUIREMENTS OF IBE SCHEME FOR DIFFERENT BASES

base	**Public key**	**Private key**	**Ciphertext**
this work $n = 1024$, $\lceil \log_2 q \rceil = 32, 32, 38, 39$			
2	32 Kbits	1,088 Kbits	1,120 Kbits
4	33 Kbits	576 Kbits	608 Kbits
512	38 Kbits	266 Kbits	304 Kbits
1024	39 Kbits	234 Kbits	273 Kbits
[22] $N = 1024$, $\lceil \log_2 q \rceil = 27$			
NA	30 Kbits	27 Kbits	30 Kbits
[41] $N = 1024$, $\lceil \log_2 q \rceil = 26$			
NA	26 Kbits	17 Kbits	31 Kbits

First, we provide storage requirements of our IBE scheme implementation and of those in [22] and [41] in Table V. We use 32-bit moduli for both bases 2 and 4 whereas we use 38 and 39-bit moduli for bases 512 and 1024, respectively. In our IBE, the public key is just a single polynomial in $\mathcal{R}_q$ as in the case of [22]. Therefore, apart from a small difference in public key sizes in Table V due to the slight difference in modulus sizes, we can claim that the public key sizes are almost the same.

Nevertheless, our scheme requires much larger storage space for user private keys and ciphertext due to the larger trapdoor size, which is proportional to the modulus size that is determined by the correctness constraint of the IBE scheme used in our implementation. The trapdoor in [22], on the other hand, is very simple but proves to be useful only in simple schemes, such as digital signatures and IBE. The figures in Table V, however, clearly show that we can compress the private key and ciphertext sizes by the factors of $1088/234 \approx 4.65$ and $1120/273 \approx 4.10$,

TABLE VI
EXECUTION TIMES OF IBE SCHEME FOR DIFFERENT BASES AND COMPARISON IN MILLISECONDS

base	**Key Gen.**	**Encryption**	**Decryption**
this work $n = 1024$, $\lceil \log_2 q \rceil = 32, 32, 37, 37$ @3.4 GHz			
2	26.65+21.80=48.45	4.03	0.31
4	11.38+12.95=24.33	1.94	0.17
512	3.75+12.34=16.09	0.87	0.11
1024	2.97+11.98=14.95	0.78	0.10
[22] $N = 1024$, $\lceil \log_2 q \rceil = 27$ @2.5 GHz			
NA	32.7	1.87	1.27
[41] $N = 1024$, $\lceil \log_2 q \rceil = 26$ @4.0 GHz			
NA	7.30 + 7.46 = 14.76	0.28	0.09

respectively, using a larger base.

We also compare our implementation with the works [22] and [41] for execution times. The time measurements in [22] are taken at a computer featuring Intel(R) Core(TM) i5-3210 CPU with a 2.50 GHz clock frequency whereas the ones in [41] were measured with Intel(R) Core(TM) i7-6700 CPU with a 4.00 GHz clock frequency without TurboBoost or hyper-threading. The implementation in [22] uses C++ as the programming language and utilizes two specialized libraries for fast arithmetic in the underlying rings and fields: NTL and GMP[3]. The one in [41], on the other hand, is written in ANSI C to have a general-purpose implementation and uses GNU GMP, similar to [22], for multi-precision arithmetic. NTL uses GMP for basic arithmetic operations whereby the latter employs highly optimized codes (e.g., assembly routines for time-critical operations). Our implementation, on the other hand, is written only in C++, uses no external library, and exploits no assembly language routines. All execution times are enumerated in Table VI.

The positive effects of using larger bases in our implementation for all three operations, namely key generation, encryption and decryption, can be observed in the execution times in Table VI. Using $b = 1024$ as opposed to $b = 2$ results in speedups of $48.45/14.95 \approx 3.24$, $4.03/0.78 \approx 5.17$, and $0.31/0.10 \approx 3.10$ in key generation, encryption, and decryption operations, respectively. The key generation operation can be performed in as low as 2.97 ms if the two-phase preimage sampling is employed. In comparison with the timing results of [22], our encryption operation is slightly slower, whereas our key generation and decryption operations outperform those in [22]. Note that our runtime results are comparable to those for the optimized implementation [41].

D. Implementation Results for CP-ABE and Comparison

To show the versatility of our trapdoor construction, we also implemented the RLWE-based CP-ABE scheme described in Section IV-C and report the implementation results in this section. We provide storage requirements for private key and ciphertext sizes and execution times for key generation, encryption and decryption operations. As the subject is relatively new, there is no lattice-based CP-ABE implementation in the literature that could be used for a fair comparison. Therefore, we use a CP-ABE implementation based on bilinear pairings in [57], which represents the state-of-the-art in the literature. Note that the comparison is, by no means, fair since the implementation in [57] is based on different security assumptions, not post-quantum and employs highly optimized code for the underlying processor hardware. Nevertheless, a comparison between the two is useful to evaluate the progress in lattice-based cryptography and assess the further effort required to close the gap in major performance indicators such as execution times.

As the authors of [57] report no storage requirement analysis, we only provide ours for user private key and ciphertext and include no comparison. The number of all attributes is the determining factor in sizes of both private key and ciphertext, whereas the latter is also affected by the number attributes in the access policy.

The formula for user private key size (in number of bits) can be given as $\ell \cdot m \cdot n \cdot \lceil \log_2 q \rceil$, where $m = \lceil \log_b q \rceil + 2$, q is the modulus, b is the base, n is the ring dimension, and ℓ is the number of attributes. The expression for ciphertext

[3]See the links http://shoup.net/ntl/ and https://gmplib.org/

TABLE VII
STORAGE REQUIREMENTS OF CP-ABE SCHEME FOR DIFFERENT BASES $n = 1024$ (IN MB)

$(\boldsymbol{\ell}, b)$	**Private key**	**Ciphertext** Minimum	Maximum
(6, 64)	1.00 / 0.21	1.16 / 0.24	1.99 / 0.41
(8, 64)	1.33 / 0.27	1.49 / 0.31	2.66 / 0.55
(16, 128)	2.73 / 0.47	2.91 / 0.50	5.47 / 0.94
(20, 128)	3.42 / 0.59	3.59 / 0.62	6.84 / 1.17
(32, 128)	5.47 / 0.94	5.64 / 0.97	10.94 / 1.88

TABLE VIII
EXECUTION TIMES OF CP-ABE SCHEME FOR DIFFERENT BASES AND COMPARISON IN MILLISECONDS

$(\boldsymbol{\ell}, b)$	**Key Generation**	**Encryption**	**Decryption**
this work @ 3.4 GHz			
(6,2)	108.07	37.50	1.54
(6,1024)	27.29	7.65	0.34
(8,2)	123.55	48.35	1.96
(8,1024)	31.20	8.91	0.42
(16,2)	217.87	95.13	3.87
(16,1024)	48.11	18.12	0.73
(20,2)	262.51	113.87	4.60
(20,1024)	56.79	22.53	0.91
(32,2)	401.92	183.42	7.26
(32,1024)	81.97	33.57	1.38
[57] adjusted for 3.4 GHz			
(6, -)	0.19	0.70	1.35
(20, -)	0.50	2.10	3.76

size is formulated as, $(2\ell - |\mathcal{W}| + 1) \cdot m \cdot n \cdot \lceil \log_2 q \rceil$, where $|\mathcal{W}|$ is the number of the attributes in the access policy. The maximum and minimum ciphertext sizes are reached for $|\mathcal{W}| = 1$ and $|\mathcal{W}| = \ell$, respectively. The storage requirements for various numbers of attributes are given in Table VII, which clearly emphasize the advantages of using a larger base.

In Table VIII, we summarize the execution times of our implementation of CP-ABE scheme along with the timings of [57]. The timings in [57] are originally given in terms of numbers of clock cycles for each iteration, which are translated here to milliseconds using 3.4 GHz as the clock frequency to match that of our computing platform. While our key generation operation is very slow compared to that in [57], it is in practical range for even relatively high numbers of attributes. Considering that it is performed infrequently (once per user), a slightly slow key generation operation can be tolerated.

Our encryption operation is also slow compared to the bilinear-pairing-based implementation in [57]. But again the execution times indicate that the scheme is practical. On the other hand, our decryption timings are faster than those in [57]. In a typical scenario, in which a CP-ABE scheme is employed, encryption operations are not performed as frequently as the decryption operation. Usually, data is encrypted once under an access policy, and decrypted multiple times by users who hold a subset of attributes that satisfies the access policy.

Finally, from throughput perspective we can even claim that our lattice-based CP-ABE scheme has certain advantages as one ciphertext encrypts four times more plaintext bits than does the pairing-based construction in [57] (1024 versus 256 as reported in [57]). As a result, our decryption operation expends 0.3 μs and 0.9 μs per bit for 6 and 20 attributes, respectively, whereas the scheme in [57] does 5.3 μs and 14.7 μs for the same operations.

In summary, we can claim that our lattice-based CP-ABE implementation is practical as far as the execution times are concerned, with the additional benefit that its security assumptions are believed to hold even in the post-quantum world.

VI. Conclusion and Future Work

In this paper, we demonstrated that Gaussian sampling for lattice trapdoors is a powerful cryptographic primitive that can be efficiently used in a diverse set of cryptographic algorithms. Our Gaussian sampling method works with arbitrary moduli, which is a requirement in majority of the cryptographic algorithms. In addition, the lattice trapdoor in our implementation can be made significantly shorter, which improves not only the storage requirements but also the execution times.

We implemented three lattice-based cryptography schemes, namely GPV signature, IBE and CP-ABE, and reported their execution times and storage requirements. We provided analyses of security and correctness constraints for all three schemes. In addition, we included security proofs for IBE and CP-ABE schemes. The implementation results confirm our claims that the three schemes can be used in practice.

Our GPV signature implementation outperforms all previous implementations based on the same trapdoor construction in every aspect. Our IBE scheme implementation provides a performance comparable to the fastest lattice-based IBE implementation in the literature, while the latter uses stronger (non-standard) assumptions. We also compared our lattice-based CP-ABE scheme with a pairing-based implementation of CP-ABE. Although our key generation and encryption operations are slower, our decryption operation yields a performance comparable to the pairing-based implementation. It should be noted that the decryption operation is expectedly executed more often than the other two operations in a CP-ABE scheme. A fast decryption operation is particularly useful when multiple users share the same access policy, such as in the case of broadcast encryption.

Finally, our implementation results are promising for the practicality of more complicated cryptographic schemes, such as KP-ABE, PE, functional encryption, and software obfuscation.

VII. Acknowledgements

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Laboratory (ARL) under Contract Numbers W911NF-15-C-0226 and W911NF-15-C-0233. The views expressed are those of the authors and do not necessarily reflect the official policy or position of the Department of Defense or the U.S. Government.

References

[1] Ajtai, M.: Generating hard instances of the short basis problem. In: ICALP. pp. 1–9 (1999)

[2] Ajtai, M.: Generating hard instances of lattice problems. Quaderni di Matematica 13, 1–32 (2004), preliminary version in STOC 1996

[3] Albrecht, M.: Lwe estimator. https://bitbucket.org/malb/lwe-estimator/ (Accessed July 2018 (commit cc5f6e8))

[4] Albrecht, M.R.: On dual lattice attacks against small-secret LWE and parameter choices in helib and SEAL. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017. Lecture Notes in Computer Science, vol. 10211, pp. 103–129 (2017), https://doi.org/10.1007/978-3-319-56614-6

[5] Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the efficacy of solving LWE by reduction to unique-svp. In: Lee, H., Han, D. (eds.) Information Security and Cryptology - ICISC 2013. Lecture Notes in Computer Science, vol. 8565, pp. 293–310. Springer (2013), https://doi.org/10.1007/978-3-319-12160-4

[6] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Mathematical Cryptology 9(3), 169–203 (2015), http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml

[7] Badawi, A.A., Polyakov, Y., Aung, K.M.M., Veeravalli, B., Rohloff, K.: Implementation and performance evaluation of rns variants of the bfv homomorphic encryption scheme. Cryptology ePrint Archive, Report 2018/589 (2018), https://eprint.iacr.org/2018/589

[8] Bansarkhani, R.E., Buchmann, J.A.: Improvement and efficient implementation of a lattice-based signature scheme. In: Lange, T., Lauter, K.E., Lisonek, P. (eds.) Selected Areas in Cryptography - SAC 2013. Lecture Notes in Computer Science, vol. 8282, pp. 48–67. Springer (2013), http://dx.doi.org/10.1007/978-3-662-43414-7

[9] Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07). pp. 321–334 (May 2007)

[10] Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2003), http://dx.doi.org/10.1137/S0097539701398521

[11] Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology - EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 533–556. Springer (2014), http://dx.doi.org/10.1007/978-3-642-55220-5

[12] Borcea, C., Gupta, A.D., Polyakov, Y., Rohloff, K., Ryan, G.W.: PICADOR: end-to-end encrypted publish-subscribe information distribution with proxy re-encryption. Future Generation Comp. Syst. 71, 177–191 (2017), https://doi.org/10.1016/j.future.2016.10.013

[13] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) 6(3), 13 (2014)

[14] Brakerski, Z., Rothblum, G.N.: Obfuscating conjunctions. J. Cryptology 30(1), 289–320 (2017), https://doi.org/10.1007/s00145-015-9221-5

[15] Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Hoffstein, J., Lauter, K., Lokam, S., Moody, D., Morrison, T., Sahai, A., Vaikuntanathan, V.: Security of Homomorphic Encryption. http://homomorphicencryption.org/white_papers/security_homomorphic_encryption_white_paper.pdf (2017)

[16] Chen, D.D., Mentens, N., Vercauteren, F., Roy, S.S., Cheung, R.C.C., Pao, D., Verbauwhede, I.: High-speed polynomial multiplication architecture for Ring-LWE and SHE cryptosystems. IEEE Transactions on Circuits and Systems I: Regular Papers 62(1), 157–166 (Jan 2015)

[17] Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex fourier series. Mathematics of computation 19(90), 297–301 (1965)

[18] Cousins, D.B., Crescenzo, G.D., Gür, K.D., King, K., Polyakov, Y., Rohloff, K., Ryan, G.W., Savaş, E.: Implementing conjunction obfuscation under entropic ring lwe. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 68–85. doi.ieeecomputersociety.org/10.1109/SP.2018.00007

[19] Cousins, D.B., Rohloff, K., Sumorok, D.: Designing an fpga-accelerated homomorphic encryption co-processor. IEEE Transactions on Emerging Topics in Computing 5(2), 193–206 (April 2017)

[20] Dai, W., Doröz, Y., Polyakov, Y., Rohloff, K., Sajjadpour, H., Savaş, E., Sunar, B.: Implementation and evaluation of a lattice-based key-policy abe scheme. IEEE Transactions on Information Forensics and Security 13(5), 1169–1184 (May 2018)

[21] Deng, H., Wu, Q., Qin, B., Domingo-Ferrer, J., Zhang, L., Liu, J., Shi, W.: Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. Inf. Sci. 275, 370–384 (2014), http://dx.doi.org/10.1016/j.ins.2014.01.035

[22] Ducas, L., Lyubashevsky, V., Prest, T.: Efficient identity-based encryption over NTRU lattices. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014. Lecture Notes in Computer Science, vol. 8874, pp. 22–41. Springer (2014), https://doi.org/10.1007/978-3-662-45608-8

[23] Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144 (2012), http://eprint.iacr.org/

[24] Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: fast-fourier, lattice-based, compact signatures over ntru (2017), submission to NIST Post-Quantum Competition

[25] Genise, N., Micciancio, D.: Faster gaussian sampling for trapdoor lattices with arbitrary modulus. Cryptology ePrint Archive, Report 2017/308 (2017), http://eprint.iacr.org/2017/308

[26] Gentry, C., Halevi, S., Smart, N.P.: Homomorphic evaluation of the AES circuit. In: CRYPTO. pp. 850–867 (2012)

[27] Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC. pp. 197–206 (2008)

[28] Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013, pp. 75–92. Springer (2013)

[29] Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015. Lecture Notes in Computer Science, vol. 9216, pp. 503–523. Springer (2015), https://doi.org/10.1007/978-3-662-48000-7

[30] Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) Automata, Languages and Programming, 35th International Colloquium, ICALP 2008. Lecture Notes in Computer Science, vol. 5126, pp. 579–591. Springer (2008)

[31] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. pp. 89–98. CCS '06, ACM, New York, NY, USA (2006)

[32] Gur, K.D., Polyakov, Y., Rohloff, K., Ryan, G.W., Savas, E.: Implementation and evaluation of improved gaussian sampling for lattice trapdoors. IACR Cryptology ePrint Archive 2017, 285 (2017), http://eprint.iacr.org/2017/285

[33] Halevi, S., Polyakov, Y., Shoup, V.: An improved rns variant of the bfv homomorphic encryption scheme. Cryptology ePrint Archive, Report 2018/117 (2018), https://eprint.iacr.org/2018/117

[34] Hanrot, G., Stehlé, D.: Worst-case hermite-korkine-zolotarev reduced lattice bases. CoRR abs/0801.3331 (2008), http://arxiv.org/abs/0801.3331

[35] Karney, C.F.F.: Sampling exactly from the normal distribution. ACM Trans. Math. Softw. 42(1), 3:1–3:14 (2016), http://doi.acm.org/10.1145/2710016

[36] Lewko, A.B., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) Advances in Cryptology - EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 62–91. Springer (2010), http://dx.doi.org/10.1007/978-3-642-13190-5

[37] Lewko, A.B., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) Advances in Cryptology - EUROCRYPT 2011. Lecture Notes in Computer Science, vol. 6632, pp. 568–588. Springer (2011), http://dx.doi.org/10.1007/978-3-642-20465-4

[38] Lindner, R., Peikert, C.: Better key sizes (and attacks) for LWE-based encryption. In: CT-RSA. pp. 319–339 (2011)

[39] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: EUROCRYPT. pp. 1–23 (2010)

[40] Lyubashevsky, V., Peikert, C., Regev, O.: A toolkit for ring-lwe cryptography. In: EUROCRYPT. vol. 7881, pp. 35–54. Springer (2013)

[41] McCarthy, S., Smyth, N., O'Sullivan, E.: A practical implementation of identity-based encryption over ntru lattices. In: O'Neill, M. (ed.) Cryptography and Coding. pp. 227–246. Springer International Publishing, Cham (2017)

[42] Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: EUROCRYPT. pp. 700–718 (2012)

[43] Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007), preliminary version in FOCS 2004

[44] Micciancio, D., Regev, O.: Lattice-based Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), https://doi.org/10.1007/978-3-540-88702-7_5

[45] Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) CCS 2007. pp. 195–203. ACM (2007)

[46] Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: CRYPTO. pp. 80–97 (2010)

[47] Peikert, C.: A decade of lattice cryptography. Foundations and Trends in Theoretical Computer Science 10(4), 283–424 (2016), https://doi.org/10.1561/0400000074

[48] Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. ACM Trans. Priv. Secur. 20(4), 14:1–14:31 (Sep 2017), http://doi.acm.org/10.1145/3128607

[49] Regev, O.: New lattice-based cryptographic constructions. J. ACM 51(6), 899–942 (2004), preliminary version in STOC 2003

[50] Regev, O.: Quantum computation and lattice problems. SIAM J. Comput. 33(3), 738–760 (2004), preliminary version in FOCS 2002

[51] Regev, O.: Lattice-based cryptography. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006. Lecture Notes in Computer Science, vol. 4117, pp. 131–141. Springer (2006), https://doi.org/10.1007/11818175_8

[52] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6), 1–40 (2009), preliminary version in STOC 2005

[53] Rückert, M., Schneider, M.: Selecting secure parameters for lattice-based cryptography. Cryptology ePrint Archive, Report 2010/137 (2010), http://eprint.iacr.org/

[54] Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer (2005), http://dx.doi.org/10.1007/11426639_27

[55] Sánchez, A.H., Rodríguez-Henríquez, F.: NEON implementation of an attribute-based encryption scheme. In: ACNS 2013. pp. 322–338 (2013)

[56] Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) Public Key Cryptography - PKC 2011. Lecture Notes in Computer Science, vol. 6571, pp. 53–70. Springer (2011), http://dx.doi.org/10.1007/978-3-642-19379-8

[57] Zavattoni, E., Perez, L.J.D., Mitsunari, S., Sánchez-Ramírez, A.H., Teruya, T., Rodríguez-Henríquez, F.: Software implementation of an attribute-based encryption scheme. IEEE Trans. Computers 64(5), 1429–1441 (2015), http://dx.doi.org/10.1109/TC.2014.2329681

[58] Zhang, J., Zhang, Z.: A ciphertext policy attribute-based encryption scheme without pairings. In: Information Security and Cryptology - Inscrypt 2011. pp. 324–340 (2011)

[59] Zhang, J., Zhang, Z., Ge, A.: Ciphertext policy attribute-based encryption from lattices. In: ASIACCS '12. pp. 16–17 (2012)