Attribute-Based Functional Encryption on Lattices
(Extended Abstract)

Xavier Boyen

December 31, 2012

Abstract

We introduce a broad lattice manipulation technique for expressive cryptography, and use it to realize functional encryption for access structures from post-quantum hardness assumptions.

Specifically, we build an efficient key-policy attribute-based encryption scheme, and prove its security in the selective sense from learning-with-errors intractability in the standard model.

[0]This is a longer version with appendices of a paper to appear at TCC 2013. Author's contact email: xb@boyen.org

1 Introduction

Attribute-Based Encryption (ABE) is a very powerful notion of encryption, where ciphertexts are not decipherable according to the ownership of a specific key (as in public-key encryption), or a specific name (as in identity-based encryption), but according to the fulfillment of a functional condition expressed as a predicate that takes multiple attributes as input.

Attribute-based encryption was first realized in a paper by Goyal *et al.* [23], although the idea was already present in the Fuzzy IBE of Sahai and Waters [34], which for the first time permitted ciphertexts to be addressed on the basis of a condition that was strictly richer than a mere equality (of keys or identities). Since then, the notion of ABE has blossomed into an entire research program known as Functional Encryption [24, 11], whereby rich functions driven by inputs from both the ciphertext and the key attempting to decrypt it, determine whether the message, or some function thereof, can be accessed. As an illustrative example of recent developments in this area, Waters very recently built a functional cryptosystem whose predicates are deterministic finite automata [36].

As impressive as these results may be, almost all of them appear to require the machinery of bilinear maps [29]—which leaves them completely vulnerable to quantum cryptanalysis, by virtue of hinging on the classically hard but quantumly easy Discrete Log problem. (Limited instances of construction from yet other techniques [16, 10] do exist, but, with assumptions that hinge on Factoring, they are equally vulnerable to quantum attacks.) With quantum computers rapidly moving from a scientific to an engineering problem, it behooves us to have safe cryptographic alternatives ready before they become a reality—possibly with nary an advance warning. Lattices appear to be our best defense, for not only are they increasingly conjectured to thwart the quantum threat in a fundamental way, they also have a rich mathematical structure that makes them well suited for building "complex" and expressive cryptographic systems.

Lattices have made their apparition in cryptography with the work Ajtai [5], and have since been used to construct a vast variety of primitives, including one-way and collision-resistant hash functions [5, 27], signatures [12, 26], public-key encryption [7, 32, 33], identity-based encryption schemes [22, 15, 1, 2], lossy trapdoor functions [31], and even a couple instances of functional encryption for inner-product [4] and threshold [3] functions. Lattices have also been very instrumental in cracking the long-standing question of realizing fully homomorphic encryption [20, 21, 14].

Lattices are indeed rapidly emerging as a mathematical platform of choice for building increasingly powerful and efficient cryptographic primitives. In addition to lattice problems being generally conjectured to withstand quantum attacks, the mathematical properties of these objects make them both relatively efficient and flexible to enable the construction of powerful cryptosystems. Research in lattice-based cryptosystems that reduce from the "Learning With Errors" (LWE) hardness assumption has been particularly active, in no small part because the average-case LWE problem is itself reducible [33, 30] from a slew of worst-case lattice problems, for a sound foundation.

Despite all of those incentives and successes, the reality is that functional encryption so far remains largely confined to the world of bilinear maps. In recent years, only a handful of such systems have been successfully realized using lattices, such as the already cited constructions of IBE [22, 1], HIBE [15, 2], IPE [4], and FuzzyIBE [3]. Further advances have remained elusive, despite the "pull" exerted by the faster pace of progress in that other world of bilinear maps. Rather disconcertingly indeed, as attempts are made to translate high-level principles of bilinear-map functional encryption into lattice analogues, serious difficulties tend to crop up in the most unexpected places when one tries to prove security. A pointed example, documented in [3], relates to the unresolved difficulties faced by those authors when trying to build ABE from LWE.

If anything, this brief history of functional encryption from lattices suggests that new ideas are in order for progress, beyond the field's classic paradigms.

1.1 Main Motivations

"Attribute-Based Encryption using Lattices" is by many authors' account an important research question, having been posed and left unanswered in an number of recent works including [15, 1, 4, 3]. Perhaps the best evidence of the problem's popularity is none other than a recent attempt by a large corporation to lay claim on its solution, in an eponymous patent application [17], even though the problem explicitly remained open to this day.[1] Why such eager enthusiasm?

First and foremost, functional encryption in general and ABE in particular are extremely powerful cryptographic constructs that would seem almost incredible—*e.g.*, by the standards of *circa* 2000. FE and ABE primarily give us unprecedented flexibility and expressiveness with which recipients can be designated in a wholesale manner. Not only do there exist direct use cases for such power (we refer to the early literature on the subject for examples), but the prospects that it opens for protocol building are highly intriguing.

As already alluded to, such rewards would be for naught if the looming threat of a catastrophic quantum cryptanalysis kept relegating it to where damage would be contained. It would be foolish to believe that because quantum registers have only grown from 5 to 7 qubits during the last decade, that their size could not suddenly become cryptographically devastating during the next one. This is where lattices come into play.

Compounding their conjectured quantum robustness, lattices also have a number of rather unique efficiency and implementation advantages. For instance, while bilinear-map cryptosystems tend to be convenient to work with on paper thanks to the availability of clean abstractions, this view hides a rather complex elliptic-curve machinery that must be securely implemented in any physical implementation. In lattice-based cryptography, the situation is reversed: schemes and proofs tend to be more complex and mired in details, but implementations require only small-number arithmetic and basic linear algebra.

Those are the reasons—from quantum peace of mind, to the sheer challenge of solving compelling theory with practical applications—why it is far from wasted effort to "reinvent" Attibute-Based Encryption, not from bilinear maps but from lattices. (And as a bonus, we introduce a new technique whose power likely reaches into FE far beyond mere ABE.)

1.2 Our Contributions

Our main result is the construction of a functional encryption scheme for monotone access structures, also known as (key-policy) attribute-based encryption, and reduce its security from LWE.

We achieve this result by way of a new lattice manipulation framework suited to the handling of complex access policies. Compared to earlier works on lattice-based IBE and FE, our framework has two distinguishing characteristics: the reliance on *ephemeral lattices* for all private-key extractions, and the subsequent application of a *basis splicing* technique which allows a recipient to convert an ephemeral lattice's basis into a basis for any lattice in a given family, as needed.

We introduce our framework in relation to a number of observations we make in our attempt to shed some light on the difficulties previously faced. This leads us to a (rather informal) discussion of FE with uniform and non-uniform policies, and how the latter appeared hard to tackle based on previous lattice techniques.

Here we focus solely on introducing our framework and building "key-policy" KP-ABE from it. We defer to future work the study of "ciphertext-policy" CP-ABE and even more ambitious FE.

[1] The US patent application [17] appears to refer to a precursor of the "Fuzzy IBE using Lattices" subsequently published in [3], wherein a superset of the authors explicitly acknolwedge that it did not extend to a proper ABE. We further opine on mathematical but not legal grounds that our ABE falls outside of the claims of [17].

2 Preliminaries

We refer to the Appendix—available in the eprint version of the paper [13]—for background on lattices in cryptography, and more specifically on Ajtai lattices, Regev's public-key encryption and the LWE assumption, discrete Gaussian sampling, preimage sampling, lattice trapdoors, and notions of noise and variance associated with the sampling algorithms.

2.1 Attribute-Based Encryption

We follow the definition of the ABE functionality as given by Goyal et al. [23], albeit for security we consider the notion of ciphertext privacy which implies both semantic security and recipient anonymity.

Definition 1 (Key-Policy Attribute-Based Encryption)**.** A Key-Policy Attribute-Based Encryption scheme consists of the following four algorithms:

Setup$(\lambda, \ell) \to (\mathsf{Pub}, \mathsf{Msk})$**:** This algorithm is input a security parameter λ and an attribute number ℓ. It outputs a public key Pub and a master key Msk.

Extract$(\mathsf{Pub}, \mathsf{Msk}, \mathsf{Policy}) \to \mathsf{Key}$**:** This algorithm takes a public key Pub, a master key Msk, and an access policy Policy. It outputs a decryption key Key.

Encrypt$(\mathsf{Pub}, \mathsf{Attrib}, \mathsf{Msg}) \to \mathsf{Ctx}$**:** This algorithm is input a public key Pub, a list of attributes Attrib, and a message bit Msg. It outputs a ciphertext Ctx.

Decrypt$(\mathsf{Pub}, \mathsf{Key}, \mathsf{Ctx}) \to b$**:** This algorithm takes a public key Pub, a decryption key Key, and a ciphertext Ctx. It outputs the bit b if the attributes Attrib used to create Ctx satisfy the policy Policy used in the creation of Key.

Definition 2 (Selective-Model KP-ABE Security)**.** A KP-ABE scheme is ciphertext-private in the selective-attribute model of security if all probabilistic polynomial time (PPT) adversaries have at most a negligible advantage in this game:

Target: The adversary declares the challenge attributes, $\mathsf{Attrib}^\dagger$, that it wishes to be challenged upon.

Setup: The challenger runs the Setup algorithm and gives the public key to the adversary.

Queries: The adversary is allowed to issue adaptive queries for private keys corresponding to policies Policy of its choice, as long as $\mathsf{Attrib}^\dagger$ does not satisfy Policy.

Challenge: The adversary signals its readiness to accept a challenge, and proposes a message to encrypt. The challenger encrypts the message for the challenge attributes $\mathsf{Attrib}^\dagger$, and then flips a random coin r. If $r = 1$, the ciphertext is given to the adversary; if $r = 0$, a random element of the ciphertext space is returned.

Queries: This is a continuation of the earlier query phase.

Guess: The adversary outputs a guess r' of r. The advantage of an adversary A in this game is defined as $|\Pr[r' = r] - \frac{1}{2}|$

One also defines an adaptive-attribute version of the above game, where the adversary may defer the choice of target attributes until requesting the challenge.

2.2 Linear Secret Sharing

Definition 3 (LSSS over $\mathbb{Z}_q$)**.** An LSSS Π over a set of parties $\mathcal{P}$ consists of an "index map" ρ and a "share-generating matrix" $\mathsf{L} \in \mathbb{Z}_q^{\ell \times \theta}$ with ℓ rows and θ columns, where ℓ is the number of shares specified by Π, and θ depends on the structure of Π. For all $i = 1, \ldots, \ell$, the function ρ maps the i-th row of L to its corresponding party. The matrix L maps an input θ-vector $\mathbf{v} = (s, r_2, \ldots, r_\theta)$, where $s \in \mathbb{Z}_q$ is the secret to be shared, and $r_2, \ldots, r_\theta \in \mathbb{Z}_q$ are random, into an output ℓ-vector $\mathsf{L}\,\mathbf{v} = (s_1, \ldots, s_\ell)$ containing the shares of the secret s according to Π. The share $s_i = (\mathsf{L}\,\mathbf{v})_i$ is assigned to party $\rho(i)$.

Every LSSS according to the above definition enjoys the linear reconstruction property. This means that if Π is an LSSS for the access structure $\mathbb{A}$, then the following is true. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\kappa_i \in \mathbb{Z}_q\}$ for $i \in I$, such that, if the $\{\lambda_i = (\mathsf{L}\,\mathbf{v})_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \kappa_i \lambda_i = s$. It was shown by Beimel [9], that these constants $\{\kappa_i\}$ can be found in time polynomial in the size of the share-generating matrix L.

Vector Secrets and Reconstruction over $\mathbb{Z}$. For the purpose of this paper, we will need a slightly modified notion of LSSS, where secrets and shares are ℓ-dimensional integer vectors in $\mathbb{Z}^\ell$, and share-generating matrices are defined over $\mathbb{Z}$ rather than over $\mathbb{Z}_q$. This creates a few issues:

1. Since secrets and shares are themselves vectors, the vector $\mathbf{v}$ of all such shares should be viewed as a tensor, and the product $(\mathsf{L} \cdot \mathbf{v})$ interpreted accordingly.
2. There is no notion of uniform share distribution over $\mathbb{Z}$: a benign issue here.
3. Reconstruction in $\mathbb{Z}$ may require fractional interpolation coefficients $\kappa_i \in \mathbb{Q}$. We alleviate this difficulty by relaxing our notion of reconstruction, allowing the reconstructed vector to be a non-zero multiple of the original vector (which is non-trivial only if the vector has dimension greater than one). Such reconstruction is possible using only integer coefficients $\kappa_i \in \mathbb{Z}$.

Low-Norm Share Generation. We will use the generic construction mechanism described in Appendix G of [25, eprint] to convert a monotone access structure into a deterministic LSSS matrix. For access formulas with AND ($\wedge$) and OR ($\vee$) gates only, it has the further advantage to build share-generating matrices $\mathsf{L} \in \{0, \pm 1\}^{\ell \times \theta}$ with ternary elements in $\{0, \pm 1\}$. For such formulas, the (unrelaxed) reconstruction coefficients κ_i will be binary in $\{0, 1\}$ by construction, even when working in $\mathbb{Z}$, hence already integer and low-norm without further relaxation.

Duplicated Attributes. For ease of exposition, we first restrict our attention to formulas where each attribute appears exactly once. Since ρ is then the identity function, we omit it from the notation altogether—until Section 4.5 and the Example Appendix of [13] where we handle missing and duplicated attributes.

3 Framework

Before delving into scheme details, it is useful to take a look at the idea behind the "basis-splicing" framework, and place it in its broader context.

3.1 Functional Encryption from Lattices

It all starts with Ajtai lattices, the Regev cryptosystem, and Gaussian sampling.

The Regev Cryptosystem. Recall that the Regev PKE scheme [33] makes use of an Ajtai lattice [5], defined as $\Lambda_q^{\perp}(\mathsf{A}) = \{\mathbf{x} : \mathsf{A}\,\mathbf{x} = \mathbf{0} \pmod q\} \subseteq \mathbb{Z}^m$, where $q \in \mathbb{Z}^+$ and $\mathsf{A} \in \mathbb{Z}_q^{n \times m}$ together specify the lattice (though not necessarily in a unique way). In Regev's PKE scheme, one assumes q fixed and $m > n \log q$. The private key is a vector $\mathbf{d} \in \mathbb{Z}^m$ with low euclidean norm $\|\mathbf{d}\| \ll q\sqrt{m}$. The public key is a pair $(\mathsf{A}, \mathbf{u})$ such that $\mathsf{A}\,\mathbf{d} = \mathbf{u} \pmod q$. To encrypt a bit $\mathrm{m} \in \{0, 1\}$, one selects a random ephemeral vector $\mathbf{s} \in \mathbb{Z}_q^n$, and output a pair $(\mathrm{c}_0, \mathbf{c}_1)$, where $\mathrm{c}_0 = \mathbf{s}^\top \mathbf{u} + \lfloor q/2 \rceil\, \mathrm{m} + \nu_0$ and $\mathbf{c}_1 = \mathbf{s}^\top \mathsf{A} + \nu_1$, and where the additive terms ν_0 and ν_1 are low-norm independent discrete gaussian noise terms. To decrypt, the private-key holder computes the difference $\Delta = \mathrm{c}_0 - \mathbf{c}_1\,\mathbf{d}$ in $\mathbb{Z}_q$, and interprets it as "$m = 1$" if (the smallest non-negative representative of the coset) Δ lies in $\{\lfloor q/4 \rceil, \lfloor 3\,q/4 \rceil\}$, and as "$m = 0$" otherwise.

Preimage Sampling. The Regev system has served as a starting point for many "expressive" functional generalizations of public-key cryptography. The key turning point in this generalization has been the development, in [22], of a "preimage sampling" technique that, given A and $\mathbf{u}$, allow one to obtain a preimage $\mathbf{d}$ such that $\mathsf{A}\,\mathbf{d} = \mathbf{u} \pmod q$ and such that $\mathbf{d}$ has the same conditional distribution given $\mathbf{u}$ as if it had been sampled first and its image computed from it. What makes the preimage-sampling approach cryptographically interesting, is that in order to sample a preimage of good quality (where the "quality" of a sample is an inverse measure of its norm), it is (conjectured) necessary to possess a good quality or low-norm basis B for the lattice $\Lambda_q^{\perp}(\mathsf{A})$. Furthermore, Ajtai's original result [5] does give us an efficient way to co-generate both a uniformly random matrix A and an associated short basis B for the lattice it induces; whereas it is a conjectured hard problem to find even a single short vector "after the fact" for a given random A. Together, these methods provide an effective way to obtain provably secure trapdoors from lattice hardness assumptions, that have been used in interesting ways to construct increasingly "expressive" functional cryptosystems: IBE [22, 1], HIBE [15, 2], IPE [4], FuzzyIBE [3], and now ABE.

More Expressive Predicates. The combination of the lattice/basis co-generation algorithm of [5], the basic public-key framework of [33], and the preimage sampling approach of [22], has led to the invention of several functional encryption schemes for various classes of functions, starting with the identity-based encryption scheme in the original paper [22]. A handful of other functional encryption schemes from lattices were later devised, including IBE in the standard model [15, 1], hierarchical IBE [15, 2], inner-product encryption [4], and fuzzy IBE [3]. At a high level, all of those schemes find their roots in the Regev PKE system, which they generalize in various ways following a common principle. The common principle is to extend Regev so that either or both the matrix A and/or the syndrome $\mathbf{u}$ depend on the functional decryption criterion, rather than being constant. In IBE, the decryption criterion is a match of identities, so we let A and/or $\mathbf{u}$ be function of the identity. In IPE and FuzzyIBE, the decryption criterion is an inner product equality or a threshold of equalities, obtained by splitting A and/or $\mathbf{u}$ into multiple shares A_i and/or $\mathbf{u}_i$, each of which depending on one of the attributes of the decryption predicate.

3.2 Complex Policies and Non-Uniformity

In our quest to understand what differentiates successes from failures in earlier lattice-based FE construction attempts, we are drawn to observe the emergence of a pattern that we shall attempt

to characterize informally (based on inductive rather than deductive reasoning).

Uniform Policies. The "successes" share a crucial simplifying characteristic: all attributes taken as formal arguments in the decryption policy are of equal importance; they play symmetrical roles.

- IBE and HIBE use trivial examples of uniform policies, because the decryption predicate is a mere equality test that treats a full identity string as a single atomic input (of variable length in the case of HIBE), comparing that of the ciphertext with that of the private key.
- IPE uses uniform policies, because none of the multiple attributes taken as inputs to the decryption predicate, plays a different role or is more important than the others. Indeed, the predicate is of the form, "$\langle \mathbf{k}, \mathbf{c} \rangle = 0 \pmod{q}$?" (where $\mathbf{k}$ and $\mathbf{c}$ are the key's and the ciphertext's attribute vectors). Now let us consider a permutation π. If we apply it to the components of $\mathbf{k}$ and also to the components of $\mathbf{c}$, one obtains the new predicate, "$\langle \pi(\mathbf{k}), \pi(\mathbf{c}) \rangle = 0 \pmod{q}$?", which is in fact unchanged and evaluates to the same value.
- FuzzyIBE uses uniform policies by same reasoning. The only difference is that here the predicate is a θ-out-of-ℓ threshold equality test between key and ciphertext attributes.

Non-Uniform Policies. To contrast, consider the following basic ABE decryption predicate: "$(A_k = A_c) \vee ((B_k = B_c) \wedge (C_k = C_c))$?" It falls within the scope of the ABE model; yet it is non-uniform since the atomic clause that takes attribute A as input, $(A_k = A_c)$, can by itself truthify the entire predicate, whereas neither the clause in B nor in C can do the same. The attributes are not symmetrical, since A carries more weight than either B or C. Per our earlier criterion, some permutations π of the attributes would not leave the predicate invariant.

Leakage from Non-Uniformity. The authors of [3] observe that the difficulty with extending existing lattice techniques into ABE stems from the conjunction of two risk factors: the necessity to prevent short-vector private keys from spilling a full basis; and the propensity of keys with asymmetrical components to do just that.

To be sure, there are examples of earlier "FE successes" that allow full-bases to be used as keys: all the HIBE schemes [15, 1, 2] fall in that category, since full bases are needed for key delegation. However, we contend that passing out full bases is not damaging in this case, because HIBE policies are trivially uniform, involving only a single attribute, so that either there is a full match or there is no match at all—no need to finesse the power of the decryption key in any way.

The other past "FE success" with multi-vector keys is the FuzzyIBE from [3]. There, a private key is a Regev key randomly secret-shared into a number of vectors function of the threshold—definitely not a full basis which would give too much power. Such sharing finesse led to an attack when one attempted to extend the scheme to ABE with non-uniform policies, because of dicrepancies in the relative importance of the private key components. *E.g.*, a key for $A \vee (B \wedge C)$ would be "heavier" at attribute A. In this situation, an adversary could, by making multiple key queries for related but distinct policies, obtain a collection of short vectors whose "heavy" coordinates together leak enough information to allow the adversary to reconstitute a "rogue" (sub-)basis. The uneven weight of the coordinates made it difficult to randomize the keys to prevent the "heavy" coordinates from leaking, without necessarily drowning the "light" coordinates in noise and render them useless.

3.3 Robust Embedding of Policies

Instead of trying to prevent the reconstitution of rogue bases from private-key vectors (which was the direction of future research envisioned in [3]), we shall make our private keys into full bases outright—albeit, bases of *ephemeral random lattices that vary with every invocation of key extraction.*

Ephemeral Lattices. Making keys from constantly changing, ephemeral lattices seems great for security—but how can such keys be useful for decryption in a Regev-like system, if the lattices used for encryption and key extraction are different? In a nutshell, the ephemeral lattices (or, rather, the Ajtai matrices defining them) will have a known structure, featuring both deterministic and randomized subcomponents. The ephemeral lattice is rather high-dimensional and its structure will encode the private-key policy attributes. The structure will allow the recipient to transform this "useless" random-lattice basis, into a basis for any target lattice, typically of a lower dimension, that belongs in a certain authorized set that corresponds to the policy encoded into the initial structure. Thus, if a private key is valid for a given ciphertext, meaning that the attributes of one satisfy the policy of the other, then the recipient is able to transform it into a basis for the lattice used in the ciphertext construction, and from there decryption à la Regev can proceed. Conversely, if a private key is invalid for a given ciphertext, the encryption lattice will be outside the authorized set, and the private key will be useless to derive a (short) basis for that lattice.

Basis Splicing. We refer as *basis splicing* the internal operations that let the recipient transform the given high-dimensional ephemeral-lattice basis, into a basis for any desired lower-dimensional lattice in the authorized set. In the case of ABE, the structure embedded in the ephemeral lattice will be obtained from an LSSS, and the basis splicing operations will amount to taking linear combinations of the basis vectors. Certain linear combinations will cause all the blinding randomness to vanish, transforming the initial *unknown* ephemeral lattice into a smaller *known* target lattice in the authorized set.

Security versus Functionality. At an intuitive level, the security benefits that we derive from our approach are twofold:

- *Private keys as full bases are more robust than single vectors.* In a system where private keys are mere vectors, there is an incentive to obtain more than one such vector, in a bid to reconctruct a rogue basis. If the key is a full basis, there is nothing to be gained in trying to obtain another, which can be generated from the first.
- *Ephemeral lattices make a very potent blinding and firewalling mechanism.* This is perhaps the most important aspect of the framework we propose: since the key-extraction mechanism involves an independently rerandomized lattice that changes upon each invocation, the private keys are in a very strong sense firewalled from one another and from the master secret.

These two properties should intuitively make it easy to construct a secure system, which should translate into easy-to-construct reductionist simulations. The main question then is to construct a scheme that can function under those parameters. This is where the basis-splicing mechanism comes into play, as we show next.

4 Scheme

As already noted, all references to the Appendix should redirect to the eprint version of this paper, available at [13].

4.1 Intuition

Setup. The system setup is very straightforward. To each (binary) attribute Attrib_i named in the system, is associated a random Ajtai matrix A_i and a matching trapdoor B_i such that $\mathsf{A}_i\,\mathsf{B}_i = \mathsf{0}$ for small $\|\mathsf{B}_i\|$. The matrices A_i form the global public key. The trapdoors B_i form the keying authority's master key.

In KP-ABE, ciphertexts are created for sets of (binary) attributes, while private keys embed the decryption policies. To make it possible to encrypt for a set of attributes, a natural idea is, for each (binary) attribute in the system, to create an Ajtai matrix A_i and an associated trapdoor T_i. The matrices A_i will form the public key; the trapdoors T_i form the master key.

Encryption. To encrypt for an attribute set $\{\mathsf{Attrib}_i\}$, one creates a matrix F by concatenating the public matrices A_i designated by the Attrib_i, filling the gaps with the zero matrix $\mathsf{0}$; one then uses F as an "encryption matrix" à la Regev. [2]

Key Extraction. To create a private key for a given decryption policy represented as an LSSS, the key-extraction authority starts by constructing a (high-dimensional) ephemeral matrix $\mathsf{M} = [\mathsf{M}_{\text{diag}}|\mathsf{M}_{\text{lsss}}]$, where M_{diag} is a block-diagonal assembly of all the A_i, and M_{lsss} is a tensor product of the LSSS matrix and a secret ephemeral randomization matrix. Using its knowledge of the master-key bases B_i, the authority creates a short basis W for the lattice $\Lambda_q^\perp(\mathsf{M})$, randomizes it into a structure-less short basis K, and returns K as the private key. Notice that the basis K is that of a fresh random lattice whose defining Ajtai matrix M is not even revealed to the recipient.

Decryption. Given a Regev ciphertext created from some encryption matrix F, the first step is to transform the private key K into a basis T for the lattice $\Lambda_q^\perp(\mathsf{F})$, using the basis-splicing technique.

The transformation requires the encryption matrix F to lie in the "span" of the (undisclosed!) ephemeral matrix M, *i.e.*, that there be a linear combination of the rows of M that yields $\mathsf{M} \hookrightarrow [\mathsf{F}|\mathsf{0}]$. By the structure of $\mathsf{M} = [\mathsf{M}_{\text{diag}}|\mathsf{M}_{\text{lsss}}]$, it follows that the i-th block-column of F is a multiple of the i-th block of M_{diag}, or, in other words, that F is the concatenation of $\mathrm{g}_i\,\mathsf{A}_i$ with computable coefficients g_i. Though K was orthogonal to M, it is not orthogonal to $[\mathsf{F}|\mathsf{0}]$. We can obtain orthogonality to $[\mathsf{F}|\mathsf{0}]$ by multiplying each row of K by an integer coefficient $\bar{\mathrm{g}}_i \propto 1/\mathrm{g}_i \pmod q$ inversely proportional modulo q to the coefficient g_i of the corresponding column of $[\mathsf{F}|\mathsf{0}]$ (taking $\bar{\mathrm{g}}_i = 0$ when corresponding to the columns of $\mathsf{0}$ or those of F associated with a coefficient $\mathrm{g}_i = 0$).

The basis K thus transformed is a matrix $[\mathsf{T}^\top|\mathsf{0}^\top]^\top$ where T has full rank and is orthogonal to F. The final observation is to take $\bar{\mathrm{g}}_i = (\prod_{j:\mathrm{g}_j\neq 0}\mathrm{g}_j)/\mathrm{g}_i$. Because those $\bar{\mathrm{g}}_i$ are already in $\mathbb{Z}$, no modular reduction is necessary to ensure that $\bar{\mathrm{g}}_i \propto 1/\mathrm{g}_i \pmod q$. Hence the norm $\|\mathsf{T}\|$ remains small when the g_i are binary or small enough. This makes of T a low-norm full-rank set, convertible into a basis suitable as a trapdoor for sampling low-norm vectors in $\Lambda_q^\perp(\mathsf{F})$.

We see that, by properly constructing M, it is possible for the recipient to know how its trapdoor K can be transformed into the desired trapdoor T, even though M itself is not revealed. Once the

[2] A Regev ciphertext $(\mathrm{c}_0, \mathbf{c}_1)$ is created in reference to an Ajtai lattice $\Lambda_q^\perp(\mathsf{F})$ defined by a known matrix F. We call the matrix F, the *Regev encryption matrix*. (It is usually denoted A but we use F to emphasize that it is a function of the encryption attributes; we reserve the notation A_i for the constant matrices in the public key.)

trapdoor T is obtained, it can be used to decrypt the ciphertext, *e.g.*, by finding a short preimage $\mathbf{d}$ of the encryption syndrome $\mathbf{u}$, *i.e.*, such that $\mathsf{F}\,\mathbf{d} = \mathbf{u} \pmod q$, and applying Regev. [3]

Issues. For this approach to work, it is necessary that the norm of the reconstructed trapdoor T be small in order to apply Regev. The only operation that can cause the norm of T to grow out of hand, is the LSSS-based derivation of T from K. In general, for circuits containing "proper" threshold gates—not just $\wedge$ nor $\vee$—with large fan-in, the coefficients g_i can become exponentially large, which would overwhelm the noise tolerance of the Regev decryption scheme unless the modulus q is itself chosen to be exponentially large.

The first good news is that, even in the pessimal case, the issue of the LSSS coefficients is somewhat mitigated by the fact that we only perform LSSS reconstruction "half-way", eschewing full-fledged Lagrange interpolation. Indeed, the worst way in which LSSS coefficients intervene in T is through simple products $\prod_j \mathrm{g}_j$—and not as ratios of products that would further require denominator elimination as, say, in the Fuzzy IBE of [3]. Intuitively, the reason why we do not need to account for—and then eliminate—the common denominator in LSSS reconstruction, is because what needs to be reconstructed is not the secret decryption itself (such as a short pre image or basis), but merely a multiple of the (public) encryption matrix F; only a multiple is needed because F induces the same Ajtai lattice as all its multiples relatively prime to q.

The second and main good news is that, as long as the only gates present are $\wedge$ and $\vee$, regardless of their size or circuit complexity, the coefficients g_i can be made binary $\in \{0, 1\}$, thereby ensuring that $\|\mathsf{T}\| \leq \|\mathsf{K}\|$. This restriction is not as severe as it looks, as it should be emphasized that circuits of $\wedge$ and $\vee$ gates already capture most cases of practical interest for (monotone) access policies. Until now, it was not known how to realize ABE involving even the simplest non-uniform policies, *e.g.*, involving only one $\wedge$ and one $\vee$ gate.

4.2 Construction

We assume the existence of the following PPT algorithms for certain lattice sampling operations. See the Appendix in [13] for some background, and the rapidly evolving literature for the fastest and tightest instantiations, *e.g.*, [18].

– $\mathsf{TrapGen}$ for co-sampling a uniform Ajtai lattice and a short basis for it [5, 6];

– $\mathsf{SampleGaussian}$ for discrete Gaussian sampling a point on a given Ajtai lattice;

– $\mathsf{SamplePreimage}$ for sampling a preimage of a given Ajtai syndrome, with a discrete Gaussian conditional density [22, 8].

– $\mathsf{ExtendRight}$ for extending a trapdoor of an Ajtai matrix A into a trapdoor of any Ajtai matrix of the form $[\mathsf{A}|\mathsf{Z}]$, as long as A has full rank [15, 1].

Remark. (Black-Box Sampling and Algorithm Parameters)
In the scheme description, we view all of the above sampling algorithms as (commodity, interchangeable) *black boxes*, without concern for their precise parameter requirements. For now, it suffices to know that the available sampling algorithms are both sufficiently fast and sufficiently tight, to make the entire system security reducible from the learning-with-error (LWE) hardness assumption with polynomially bounded parameters, so that is can in turn be further (quantumly [33], or for large moduli classically [30]) reduced from worst-case lattice assumptions.

The KP-ABE scheme consists of four algorithms specified as follows.

[3] Because the private key is a full basis, it allows the recipient to find a preimage for any syndrome; hence the encryption syndrome $\mathbf{u}$ may change with each ciphertext.

kpABE.Setup$(1^\lambda, 1^\ell)$**:** Given a security parameter λ, and an attribute bound ℓ:

1. Select a security dimension $n > \Omega(\lambda)$ and a base lattice dimension $m > 2\,n\,\log q$, together with a prime modulus $q > 2$. (See the Appendix for the constraints on q in function of the desired tightness α of LWE—the larger the modulus, the weaker the assumption.)
2. Use algorithm $\mathsf{TrapGen}(1^\lambda)$ to select, for each $i \in [\ell]$, a uniformly random $n \times m$-matrix $\mathsf{A}_i \in \mathbb{Z}_q^{n\times m}$ with a full-rank m-vector set $\mathsf{B}_i \subseteq \Lambda_q^\perp(\mathsf{A}_i)$ that satisfies a low-norm condition.
3. Select a uniformly random $n \times m$-matrix $\mathsf{A}_0 \in \mathbb{Z}_q^{n\times m}$.
4. Select a uniform random n-vector $\mathbf{u} \in \mathbb{Z}_q^n$.
5. Output the public key and master key,

$$\mathsf{Pub} \;=\; \Big(\; \{\mathsf{A}_i\}_{i\in[\ell]},\; \mathsf{A}_0,\; \mathbf{u} \;\Big) \qquad ; \qquad \mathsf{Msk} \;=\; \Big(\; \{\mathsf{B}_i\}_{i\in[\ell]} \;\Big)$$

kpABE.Extract$(\mathsf{Pub}, \mathsf{Msk}, \mathsf{Policy})$**:** On input a public key denoted Pub, a master key denoted Msk, and an access structure denoted Policy, do:

1. Convert Policy into a (low-norm, and preferably deterministic) Linear Span Program matrix $\mathsf{L} \in \mathbb{Z}^{\ell\times(1+\theta)}$, assigning the i-th row of L to the binary attribute of index $i \in [\ell]$. The columns $j \in [0, \theta]$ are numbered from 0 to θ, with $\theta \le \ell$ being a function of Policy. The linear encoding rule we adopt for L is that, for a binary attribute list represented as $\mathsf{Attrib} \in \{0,1\}^\ell$ or $\mathsf{Attrib} \subseteq [\ell]$, the (monotone) access policy is satisfied iff the rows of L selected by Attrib contain in their span the row-vector $\big[1, 0, \ldots, 0\big] \in \mathbb{Z}^{1+\theta}$.
2. Select θ ephemeral uniform random $n \times m$-matrices $\mathsf{Z}_j \in \mathbb{Z}_q^{n\times m}$ for $j \in [\theta]$.
3. Construct a "virtual encryption matrix" $\mathsf{M} \in \mathbb{Z}_q^{\ell\,n\times(\ell+1+\theta)\,m}$, consisting of $\ell \times (\ell+1+\theta)$ blocks of $n \times m$-"sub-matrices", by translating the sharing matrix $\mathsf{L} = \big(l_{i,j}\big)_{i\in[\ell],j\in[1+\theta]}$ as follows,

$$\mathsf{M} \;=\; \left[\begin{array}{c|c|c} \begin{array}{ccc} \boxed{\mathsf{A}_1} & & \\ & \boxed{\mathsf{A}_2} & \\ & & \ddots \\ & & \boxed{\mathsf{A}_\ell} \end{array} & \begin{array}{c} l_{1,0}\,\boxed{\mathsf{A}_0} \\ l_{2,0}\,\boxed{\mathsf{A}_0} \\ \vdots \\ l_{\ell,0}\,\boxed{\mathsf{A}_0} \end{array} & \begin{array}{ccc} l_{1,1}\,\boxed{\mathsf{Z}_1} & \ldots & l_{1,\theta}\,\boxed{\mathsf{Z}_\theta} \\ l_{2,1}\,\boxed{\mathsf{Z}_1} & \ldots & l_{2,\theta}\,\boxed{\mathsf{Z}_\theta} \\ \vdots & & \vdots \\ l_{\ell,1}\,\boxed{\mathsf{Z}_1} & \ldots & l_{\ell,\theta}\,\boxed{\mathsf{Z}_\theta} \end{array} \\ \underbrace{\qquad\qquad}_{\text{Public, constant, from }\mathsf{Pub}} & \underbrace{\qquad}_{\text{From }\mathsf{Pub}} & \underbrace{\qquad\qquad\qquad}_{\text{Secret, random, ephemerals}} \end{array}\right] \bmod q$$

   Each row of L maps to a particular attribute according to the map ρ associated with the secret-sharing scheme. In this section, we are assuming for simplicity that each attribute (of index $\#i$) appears exactly once (on the i-th row), making ρ the identity function. This restriction is lifted in Section 4.5, to handle missing and duplicated attributes.
4. Build a "structureless" random trapdoor K for $\Lambda_q^\perp(\mathsf{M})$, thus satisfying $\mathsf{M}\cdot\mathsf{K} = \mathsf{0} \pmod q$. This can be done using $\mathsf{ExtendRight}$, based on the fact that $\mathsf{M} = [\mathsf{M}_{\text{trapdoor}}|\mathsf{M}_{\text{extension}}]$, where $\mathsf{M}_{\text{trapdoor}} = \mathit{Diag}(\mathsf{A}_1, \ldots, \mathsf{A}_\ell)$ has full rank and a trivial trapdoor $\mathit{Diag}(\mathsf{B}_1, \ldots, \mathsf{B}_\ell)$.

   Unless $\mathsf{ExtendRight}$ is already guaranteed to produce an extended basis W whose vectors are idenpendently and identically distributed, it is necessary to rerandomize it to achieve this condition. Let K be the resulting "structureless" trapdoor for M.

5. A redundant form of the policy-based private key may be output, as,

$$\mathsf{Key} \;=\; \Big(\; \mathsf{K},\; \mathsf{L} \;\Big)$$

However, two optimizations can be made:

(a) If the sharing matrix L is deterministic in Policy, it may be omitted.

(b) It is not necessary to transmit all of K since the decryptor will only ever need the upper-left quadrant of dimension $(\ell+1)\,m \times (\ell+1)\,m$, which we denote by $\mathsf{K}' \in \mathbb{Z}^{(\ell+1)\,m \times (\ell+1)\,m}$.

Hence, the private key for Policy may be given in compressed form, as,

$$\mathsf{Key} \;=\; \boxed{\mathsf{K}'}$$

kpABE.Encrypt$(\mathsf{Pub}, \mathsf{Attrib}, \mathsf{Msg})$**:** On input a public key Pub, an attribute list $\mathsf{Attrib} \subseteq [\ell]$, and a message bit $\mathsf{Msg} \in \{0,1\}$, do:

1. Assemble an "encryption matrix" $\mathsf{F} \in \mathbb{Z}_q^{n \times (\ell+1)\,m}$, obtained as the concatenation of, for each $i \in [\ell]$, either A_i if $i \in \mathsf{Attrib}$, or $\mathsf{0}$ if $i \notin \mathsf{Attrib}$, and A_0, as follows,

$$\mathsf{F} \;=\; \left[\begin{array}{c|c|c|c} \mathsf{F}_1 \doteq & & \mathsf{F}_\ell \doteq & \mathsf{F}_0 \doteq \\ \boxed{\mathsf{A}_1} & \cdots & \boxed{\mathsf{A}_\ell} & \boxed{\mathsf{A}_0} \\ \underbrace{\text{or } \mathsf{0}}_{\mathsf{A}_i \text{ included iff } i \in \mathsf{Attrib}} & & \text{or } \mathsf{0} & \end{array}\right]$$

2. Select a uniform random n-vector $\mathbf{s} \in \mathbb{Z}_q^n$.

3. Select a low-norm Gaussian noise scalar $\nu_0 \in \mathbb{Z}$ according to some parametric distribution Ψ_α (see Appendix), and compute the scalar,

$$\mathrm{c}_0 \;=\; \big(\; \mathbf{s}^\top \cdot \mathbf{u} + \nu_0 + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{Msg} \;\big) \bmod q$$

4. Select a low-norm Gaussian noise vector $\nu_1 \in \mathbb{Z}^{(\ell+1)\,m}$ whose components are identically and independently distributed from Ψ_α, and compute the vector,

$$\mathbf{c}_1 \;=\; \big(\; \mathbf{s}^\top \cdot \mathsf{F} + \nu_1 \;\big) \bmod q$$

5. Output the ciphertext,

$$\mathsf{Ctx} \;=\; \Big(\; \mathrm{c}_0,\; \mathbf{c}_1 \;\Big)$$

(It is not necessary to transmit the components of $\mathbf{c}_1$ that contain only added ν_1-noise, i.e., we only need to transmit the components of $\mathbf{c}_1$ at coordinates where $\mathsf{F}_i \neq 0$.)

kpABE.Decrypt$(\mathsf{Pub}, \mathsf{Key}, \mathsf{Ctx})$**:** Given a public key Pub, a policy-based key Key (for known policy Policy), and a ciphertext Ctx (for known attributes Attrib):

1. Find an as-short-as-feasible ℓ-vector $\mathbf{g} \in \mathbb{Z}^\ell$ satisfying the two conditions:

$$\mathbf{g}^\top \cdot \mathsf{L} = [d, 0, \ldots, 0] \propto [1, 0, \ldots, 0] \qquad ; \qquad \forall i \in [\ell] : \ (g_i = 0) \vee (i \in \mathsf{Attrib})$$

   Namely, one finds a linear combination of the rows of L that yields some small d-multiple of $[1, 0, \ldots, 0]$ with $d \in \mathbb{Z} \setminus \{0\}$, using only rows corresponding to attributes in Attrib. This is possible iff Attrib satisfies Policy.

2. Notionally apply the linear combination $\mathbf{g}$ to the "block-rows" of M, to transform the "virtual" encryption matrix M into a "real" encryption matrix M' that matches the encryption matrix F of the given ciphertext (up to constant factors):

$$\mathsf{M}' \ = \ \left[\begin{array}{c|c|c|c|c|c} g_1 \boxed{\mathsf{A}_1} \text{ or } \mathsf{0} & g_2 \boxed{\mathsf{A}_2} \text{ or } \mathsf{0} & \ldots & g_\ell \boxed{\mathsf{A}_\ell} \text{ or } \mathsf{0} & d \cdot \boxed{\mathsf{A}_0} & \underbrace{\mathsf{0} \cdot \mathsf{Z}_1 \mid \ldots \mid \mathsf{0} \cdot \mathsf{Z}_\theta}_{\mathsf{0}} \end{array} \right] \bmod q$$

   This is defined, even though the decryptor does not know the Z_i, for they all cancel out.

3. Let M'' be the matrix containing only the $|\mathsf{Attrib}| + 1$ non-zero "block-columns" of M' as shown above. Let K'' be the matrix obtained by removing from K the matching rows and columns—i.e., rows and columns with the same indices as the columns removed from M'. (Dimension-wise, we obtain $\mathsf{M}'' \in \mathbb{Z}_q^{n \times (|\mathsf{Attrib}|+1)\, m}$ and $\mathsf{K}'' \in \mathbb{Z}^{(|\mathsf{Attrib}|+1)\, m \times (|\mathsf{Attrib}|+1)\, m}$.) We have $\mathsf{M}' \cdot \mathsf{K} = \mathsf{0}$; therefore $\mathsf{M}'' \cdot \mathsf{K}'' = \mathsf{0}$, and K'' is a short basis of $\Lambda_q^\perp(\mathsf{M}'')$.

4. Likewise, let F'' be the matrix retaining the $|\mathsf{Attrib}|+1$ non-zero "block-columns" of F; and let $\mathbf{c}_1''$ be the ciphertext vector from which only the matching components of $\mathbf{c}_1$ remain.

5. We now build a trapdoor for the encryption matrix F, or, rather, its reduced form F''. Let $\mathsf{1}$ be the $m \times m$ identity matrix, and define the diagonal matrices,

$$\mathsf{G} \ = \ \begin{bmatrix} g_1 \cdot \boxed{\mathsf{1}} & & & \\ & \ddots & & \\ & & g_\ell \cdot \boxed{\mathsf{1}} & \\ & & & d \cdot \boxed{\mathsf{1}} \end{bmatrix} \quad ; \quad \mathsf{G}'' \ = \ \begin{bmatrix} \text{non-zero} \\ \text{diagonal} \\ \text{blocks} \\ \text{of } \mathsf{G} \end{bmatrix} \in \mathbb{Z}^{(|\mathsf{Attrib}|+1)\, m \times (|\mathsf{Attrib}|+1)\, m}$$

   Notice $\mathsf{F}'' \cdot \mathsf{G}'' = \mathsf{M}'' \pmod q$. Since $\mathsf{M}'' \cdot \mathsf{K}'' = \mathsf{0} \pmod q$, we have $\mathsf{F}'' \cdot \mathsf{G}'' \cdot \mathsf{K}'' = \mathsf{0} \pmod q$. Compute $\mathsf{T}'' = \mathsf{G}'' \cdot \mathsf{K}''$, whose norm is bounded as $\|\mathsf{T}''\| \le \|\mathsf{G}''\| \, \|\mathsf{K}''\| \le \max\{g_i, d\} \, \|\mathsf{K}\|$. The result T'' is our desired trapdoor for sampling short vectors in $\Lambda_q^\perp(\mathsf{F}'')$.

6. Using $\mathsf{SamplePreimage}$ with trapdoor T'', find a short solution $\mathbf{f}''$ of $\mathsf{F}'' \cdot \mathbf{f}'' = \mathbf{u} \pmod q$.

7. Compute $\mathrm{v} = \mathrm{c}_0 - (\mathbf{f}'')^\top \cdot \mathbf{c}_1'' \bmod q$, and represent its coset as an integer $v \in [-\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor]$.

8. Output the decrypted message bit as,

$$b \ = \ \begin{cases} 0 & \text{if } \|v\| \le \lfloor \frac{q}{4} \rfloor \\ 1 & \text{if } \|v\| \ge \lceil \frac{q}{4} \rceil \end{cases}$$

4.3 Correctness

Theorem 4. *For usual values of the lattice parameters in Regev-like encryption systems, the key-policy attribute-based encryption scheme of the previous section will correctly decrypt authorized ciphertexts with overwhelming probability.*

Proof. To see this, suppose that the "independent" initial bases and short vectors (namely, B_i, Y_i, $\mathbf{e}_{i,j}$, $\mathbf{d}_{i,j}$) are sampled with a suitable Gaussian parameter σ, for instance using the tools from [22, 8]. Then, the norm of all "dependent" bases and vectors that are supposed to be short, will be bounded by multiples of σ to which certain "growth coefficients" will have applied. To bound those, we note that the only processes in the whole system that will induce "growth", are:

- in $\mathsf{Extract}$: the randomized invocation of $\mathsf{ExtendRight}$ to obtain K, which merely multiplies the norm of the master-key trapdoors by a constant factor independent of the data;
- in $\mathsf{Decrypt}$: the calculation of the trapdoor T'' from K'', which as we already noted multiplies the norm of K'' by a factor $\leq \max\{g_i, d\}$ that only depends on the linear-sharing reconstruction vector $\mathbf{g}$, itself function of the function Policy and its inputs Attrib.

Bounding $\max\{g_i, d\}$ for access-structure circuits with many gates can be tedious, but we note that $\max\{g_i, d\}$ will be dominated by the presence of large threshold gates. On the contrary, $\wedge$ and $\vee$ gates are essentially harmless, as shown below.
Claim. For a circuit consisting only of $\wedge$ and $\vee$ gates, $\max\{g_i, d\} = 1$.

Proof. There exists a deterministic construction of a linear sharing matrix L that guarantees binary reconstruction coefficients in this case (see Preliminaries). □

We defer to the full paper the exact quantification of the various norm and noise parameters. Of course, while the growing norm of supposedly short vectors can be compensated by commensurately increasing the modulus q, this is best avoided for efficiency reasons. □

4.4 Security

Theorem 5. *If there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ with advantage $\epsilon > 0$ in a selective-security key-policy attack against the above scheme, then there exists a probabilistic polynomial-time algorithm $\mathcal{B}$ that decides the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-LWE problem with advantage $\epsilon/2$, where $\alpha = O(poly(n))$.*

Proof. In the LWE problem, the decision algorithm is given access to a sampling oracle, $\mathcal{O}$, which is either a pseudo-random sampler $\mathcal{O}_s$ with embedded secret $s \in \mathbb{Z}_q^n$, or a truly random sampler $\mathcal{O}_\$$. Our decider algorithm $\mathcal{B}$ will simulate an attack environment for, and exploit the prowesses of $\mathcal{A}$, to decide which oracle it is given. The reduction proceeds as follows.

Instance. $\mathcal{B}$ requests from $\mathcal{O}$ and obtains $((1+\ell)\,m+1)$ LWE samples that we denote as,

$$\begin{aligned}
\left[\ (\mathbf{w}_{-1}, \mathrm{v}_{-1})\ \right] &\in (\mathbb{Z}_q^n \times \mathbb{Z}_q) \\
\left[\ (\mathbf{w}_0^1, \mathrm{v}_0^1), \ldots, (\mathbf{w}_0^m, \mathrm{v}_0^m)\ \right] &\in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^m \\
\left[\ (\mathbf{w}_1^1, \mathrm{v}_1^1), \ldots, (\mathbf{w}_1^m, \mathrm{v}_1^m)\ \right] &\in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^m \\
&\vdots \\
\left[\ (\mathbf{w}_\ell^1, \mathrm{v}_\ell^1), \ldots, (\mathbf{w}_\ell^m, \mathrm{v}_\ell^m)\ \right] &\in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^m
\end{aligned}$$

Target. $\mathcal{A}$ announces a target attribute vector, denoted $\mathsf{Attrib}^\dagger$, on which it wishes to be challenged.

Setup. $\mathcal{B}$ constructs the public key Pub as follows:

1. The vector $\mathbf{u} \in \mathbb{Z}_q^n$ is constructed from the LWE samples of index -1: simply set $\mathbf{u} = \mathbf{w}$.

2. The matrix $\mathsf{A}_0 \in \mathbb{Z}_q^{n \times m}$ is built from the LWE samples of index 0: set $\mathsf{A}_0 = [\mathbf{w}_0^1 | \dots | \mathbf{w}_0^m]$.
3. For each $i \in [\ell]$ such that attribute $i \in \mathsf{Attrib}^\dagger$, the matrix A_i is constructed from the LWE samples of index i in a similar way as above: for $i \in \mathsf{Attrib}^\dagger$, set $\mathsf{A}_i = [\mathbf{w}_i^1 | \dots | \mathbf{w}_i^m]$.
4. For each $i \in [\ell]$ such that attribute $i \notin \mathsf{Attrib}^\dagger$, the matrix A_i is constructed as in the real scheme using $\mathsf{TrapGen}$, which provides an associated low-norm full-rank matrix B_i such that $\mathsf{A}_i \cdot \mathsf{B}_i = \mathsf{0}$. (The LWE samples of all indices $i \notin \mathsf{Attrib}^\dagger$ will remain unused.)

The resulting public key Pub is given to $\mathcal{A}$.

Queries. $\mathcal{A}$ is allowed to make adaptive queries for keys Key for policies Policy that the target attribute list $\mathsf{Attrib}^\dagger$ does not satisfy. $\mathcal{B}$ constructs and returns a key Key for each query Policy, as follows.

1. As in the real scheme, derive from Policy a (low-norm) linear sharing matrix $\mathsf{L} \in \mathbb{Z}^{\ell \times (1+\theta)}$.
2. Let $\phi = |\mathsf{Attrib}^\dagger|$. Make L' from L, keeping only the rows of index i such that $i \in \mathsf{Attrib}^\dagger$. Make L'' from L' by dropping the leftmost column of index $j = 0$ (keeping $j = 1, \dots, \theta$).
3. W.l.o.g., suppose that $\mathsf{Attrib}^\dagger = \{i_1, i_2, \dots, i_\phi\} = \{1, 2, \dots, \phi\}$; i.e., the first ϕ attributes, from 1 to ϕ, are arbitrarily assumed to be the attacker's targets.
4. W.l.o.g., suppose that the ϕ left-most columns of L'' form a ϕ-dimensional square matrix of full rank. The columns of L from which L'' is derived can always be reordered to achieve this, since the order of its columns (other than that of index $j = 0$) is arbitrary. Notice that this step requires that the challenge $\mathsf{Attrib}^\dagger$ do *not* satisfy the query Policy. If it did, by definition some non-zero $[d, 0, \dots, 0]^\top$ would be in the span of L, and thus $[0, \dots, 0]^\top$ non-trivially in that of L''; therefore the ϕ left-most columns of L'' would not be full-rank.
5. Invoking $\mathsf{TrapGen}$, sample ϕ random matrices $\mathsf{Z}_i \in \mathbb{Z}_q^{n \times m}$ with short bases $\mathsf{Y}_i \in \mathbb{Z}^{m \times m}$, for all $i \in \mathsf{Attrib}^\dagger$ (*i.e.*, w.l.o.g., $i = 1, \dots, \phi$ are the indices of the Z_i with trapdoor Y_i).
6. Build a "virtual encryption matrix" M exactly as in the real scheme (see below about the boxes), as,

$$\mathsf{M} = \left[\begin{array}{ccccc|c|ccc} \mathsf{A}_1 & & & & & l_{1,0}\,\mathsf{A}_0 & l_{1,1}\,\mathsf{Z}_1 & \dots & l_{1,\phi}\,\mathsf{Z}_\phi & \dots & l_{1,\theta}\,\mathsf{Z}_\theta \\ & \ddots & & & & \vdots & \vdots & & \vdots & & \vdots \\ & & \mathsf{A}_\phi & & & l_{\phi,0}\,\mathsf{A}_0 & l_{\phi,1}\,\mathsf{Z}_1 & \dots & l_{\phi,\phi}\,\mathsf{Z}_\phi & \dots & l_{\phi,\theta}\,\mathsf{Z}_\theta \\ & & & \boxed{\ddots} & & \vdots & \vdots & & \vdots & & \\ & & & & \boxed{\mathsf{A}_\ell} & l_{\ell,0}\,\mathsf{A}_0 & l_{\ell,1}\,\mathsf{Z}_1 & \dots & l_{\ell,\phi}\,\mathsf{Z}_\phi & \dots & l_{\ell,\theta}\,\mathsf{Z}_\theta \end{array}\right] \bmod q$$

7. Denote by Z the $(\phi n \times \phi m)$-submatrix of M made of the blocks $l_{j,i}\,\mathsf{Z}_i$ whose $i, j \in [\phi]$. Per Lemma 6, we can build (from the Y_i) a single trapdoor Y for Z as a whole.

   Lemma 6. *For $i = 1, \dots, \phi$, let $\mathsf{Z}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathsf{Y}_i \in \mathbb{Z}^{m \times m}$ such that $\mathsf{Z}_i\,\mathsf{Y}_i = \mathsf{0} \pmod q$. Suppose also that each Y_i is a basis of $\Lambda_q^\perp(\mathsf{Z}_i)$ and has low norm $\|\mathsf{Y}_i\| \le \beta \in \mathbb{R}$. Define,*

$$\mathsf{Z} = \begin{bmatrix} l_{1,1}\,\mathsf{Z}_1 & \cdots & l_{1,\phi}\,\mathsf{Z}_\phi \\ \vdots & \ddots & \vdots \\ l_{\phi,1}\,\mathsf{Z}_1 & \cdots & l_{\phi,\phi}\,\mathsf{Z}_\phi \end{bmatrix} \bmod q$$

Then, for any full-rank integer matrix $\left(l_{i,j}\right)$ *with* $i, j \in [\phi]$*, the Ajtai lattice induced by* $\mathsf{Z} \in \mathbb{Z}_q^{\phi n \times \phi m}$ *admits an efficiently computable (in fact constant) trapdoor* $\mathsf{Y} \in \mathbb{Z}^{\phi m \times \phi m}$ *i.e., such that* Y *is a basis of* $\Lambda_q^{\perp}(\mathsf{Z})$ *with bounded norm* $\|\mathsf{Y}\| \leq \beta$*.*

Proof. Take,

$$\mathsf{Y} \;=\; \begin{bmatrix} \mathsf{Y}_1 & & 0 \\ & \ddots & \\ 0 & & \mathsf{Y}_\phi \end{bmatrix}$$

We have that $\mathsf{Z} \cdot \mathsf{Y} = \mathsf{0} \pmod q$, that Y is a basis for $\Lambda_q^{\perp}(\mathsf{Z})$, and that $\|\mathsf{Y}\| \leq \max_i \|\mathsf{Y}_i\|$. □

8. Observe that we now have a trapdoor for every lattice defined by a submatrix of M encased in one of the boxes shown in Step 6. Let us notionally reorder the columns of M by swapping the ϕ left-most A_i-block-columns with the ϕ left-most Z_i-block-columns. We get a matrix $\mathsf{M}' = [\mathsf{M}'_{\text{trapdoor}} | \mathsf{M}'_{\text{extension}}]$, where $\mathsf{M}'_{\text{trapdoor}}$ is full-rank, block-diagonal, and each of its blocks has an associated trapdoor. We can thus trivially build a trapdoor for all of $\mathsf{M}'_{\text{trapdoor}}$. By invoking $\mathsf{ExtendRight}$, we extend this into a trapdoor W' for all of M'. Reordering the rows of W' yields a trapdoor for the original M above: call it W.
9. Randomize W into a structure-less basis K whose norm matches that of the real scheme. (This step is only necessary if $\mathsf{ExtendRight}$ does not already produce a basis whose vectors all have the target discrete Gaussian distribution already; if they do, let $\mathsf{K} = \mathsf{W}$.)

This concludes the simulation of the private-key extraction. The adversary $\mathcal{A}$ is given the resulting $\mathsf{Key} = \left(\mathsf{K}, \mathsf{L}\right)$. Notice that it has exactly the same distribution as in the real scheme.

Challenge. $\mathcal{A}$ signals that it is ready to accept a challenge, and chooses a message bit $\mathsf{Msg}^\dagger \in \{0, 1\}$. $\mathcal{B}$ responds with a ciphertext $\mathsf{Ctx}^\dagger = \left(\mathrm{c}_0^\dagger, \mathbf{c}_1^\dagger\right)$ assembled from the LWE instance, as follows:

1. Let $\mathrm{c}_0^\dagger = \mathrm{v}_{-1} + \lfloor \frac{q}{2} \rfloor \cdot \mathsf{Msg}^\dagger$.
2. Let $\mathbf{c}_1^\dagger = \big[\; \underbrace{\mathrm{v}_1^1, \ldots \mathrm{v}_1^m}_{\text{if } 1 \in \mathsf{Attrib}^\dagger}, \ldots, \underbrace{\mathrm{v}_\ell^1, \ldots \mathrm{v}_\ell^m}_{\text{if } \ell \in \mathsf{Attrib}^\dagger}, \underbrace{\mathrm{v}_0^1, \ldots \mathrm{v}_0^m}_{\text{always}} \;\big]$

Observe that when the v_i come from a genuine LWE oracle, the foregoing is a well-formed Regev-like encryption of $\mathsf{Msg}^\dagger$ for the encryption matrix F indicated by the challenge $\mathsf{Attrib}^\dagger$. On the contrary, when the v_i come from a random fake LWE oracle, the ciphertext is independent of the message bit since $\mathrm{c}_0^\dagger$ in particular is uniformly and independently distributed.

Continuation. $\mathcal{A}$ is allowed to continue making further private-key extraction queries, after having obtained the challenge ciphertext.

Decision. $\mathcal{A}$ eventually emits a guess, whether $\mathsf{Ctx}^\dagger$ was actually a valid encryption of $\mathsf{Msg} \in \{0, 1\}$ as requested. $\mathcal{B}$ uses the guess to decide whether the LWE oracle $\mathcal{O}$ was genuine. If $\mathcal{A}$ says "valid", then $\mathcal{B}$ says "*genuine*"; if $\mathcal{A}$ says "invalid", then $\mathcal{B}$ says "*fake*".

If the adversary succeeds in guessing $\mathsf{Msg}^\dagger$ with probability at least $\frac{1}{2} + \epsilon$, then our decision algorithm $\mathcal{B}$ will correctly guess the nature of the LWE oracle with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$. This concludes the proof of the security reduction. □

4.5 Extensions

So far we have assumed, merely for simplicity of notation, that policies will only encode monotone access structures given as formulas where each attribute appears as argument exactly once. We now show how to list such limitations.

4.5.1 Duplicated Attributes

Arbitrary monotone policies will generally be expressed as formulas where various attributes appear zero, once, or even multiple times. Accordingly, we show how to handle policies that can comport arbitrarily many $\wedge$ and $\vee$ gates, and an arbitrary wiring of the attribute inputs to feed them, including duplication. [4] The idea is very simple:

kpABE.Setup' is unchanged from the original version: to each attribute one continues to associate one Ajtai matrix A_i and its trapdoor B_i.

kpABE.Setup' also remains the same: the ciphertext is constructed as before, around a Regev encryption matrix F that either includes or excludes each submatrix A_i depending on whether or not the respective attribute $i \in \mathsf{Attrib}$.

kpABE.Extract' must be modified to allow for duplicate occurrences of the same attribute in the Boolean expression of Policy. This is done as follows:

1. Give each occurrence of some attribute $\#i$ in Policy a unique label, say $\#i.1$ and $\#i.2$, and accordingly rewrite the policy Policy into Policy' as a function of the augmented attributes. Policy' has the same topology (structure and size) as Policy, but its input literals are now unique. Keep track of the mapping from the augmented attributes i' to the original attributes i by means of a surjective map $\rho : i' \mapsto i$.
2. Construct the sharing matrix L in the regular way from the augmented-attribute formula Policy'. For each original attribute $\#i$, there will be as many rows in L as the number of occurrences of $\#i$ in the original Policy.
3. Construct the "virtual encryption matrix" M from L as before. Since the augmented attributes that emanate from the same original attribute, all refer to the same public matrix A_i, the key-extraction matrix M will thus contain multiple copies of A_i, albeit on different columns.

Once M has been constructed with possibly duplicated A_i on its left-side block-diagonal, key extraction both in the real scheme and in the simulation will proceed as usual. The only effect of the duplication is that, in the simulation, knowledge of trapdoors B_i will be linked to the presence of the original attributes—not the augmented ones—in $\mathsf{Attrib}^\dagger$.

kpABE.Decrypt' requires a small adjustment to cope with duplicated attributes in the Policy encoded in the decryption key. Essentially, before applying the decryption algorithm, the decryptor needs to avail himself as many copies of the attribute as he will need. This is done by duplicating the various fragments of $\mathbf{c}_1$ that correspond to the attributes that need to be duplicated, before using the result in the normal decryption process.

[4] We must however continue to caution on the use of t-out-of-n threshold gates $\geq_t$, because unless $t = 1$ or $t = n$ we cannot guarantee in general that the LSSS matrix L and the reconstruction coefficients will be small. Fortunately, as long as repeated attribute inputs are allowed, every possible monotone access structure can be expressed using only $\vee$ and $\wedge$ gates, in such a way that L is a binary or ternary matrix.

This construction is very efficient as the ciphertext size remains unchanged in $|\mathsf{Attrib}|$, and the private key size has the same dependency on $|\mathsf{Policy}|$ as it did without attribute duplication (of course, $|\mathsf{Policy}|$ can now grow arbitrarily).

4.5.2 Negated Attributes

Handling negation is more complex. One approach is to pair each attribute with its logic opposite, and let ciphertexts and policies be expressed as functions of the positive and negative literals. However, this does not quite yet capture the richness of formulas that allow an unrestricted use of $\neg$-gates, in addition to $\vee$-gates and $\wedge$-gates

One possibility is to generalize the positive/negative dual representation trick at every level of the formula tree, ensuring that to every node is associated two values that are the logical complement of each other.

This is all we are going to say about this topic. We leave it as an open problem the task of properly extending our scheme to handle negation in an efficient way.

4.5.3 Elimination of A_0

The astute reader will observe that the public matrix A_0 never uses any trapdoor, in either the scheme or the proof.

A_0 is not essential, and can be eliminated if we move, in $\mathsf{Extract}$'s construction of M, the LSSS coefficients $l_{i,0}$ from A_0 to the A_i. Some care must then be taken with the application of $\mathsf{ExtendRight}$, since the left-hand block-diagonal part of M may no longer have full rank, but this can be handled.

Aside from making the keys and ciphertexts marginally smaller, the main benefit of eliminating A_0 is to remove the coefficient d from the expression of G in the $\mathsf{Decrypt}$ algorithm, thus possibly lowering the norm of the reconstructed decryption trapdoor T'' in the general case where threshold gates are present. (The norm of T'' is never an issue if only $\vee$ and $\wedge$ gates are present.)

5 Conclusion

In this paper, we have introduced a new cryptographic framework for performing complex lattice basis manipulations, of the kind that seemingly can unlock the construction of very powerful and expressive cryptosystems such as functional encryption. We demonstrated its power and flexibility by building the first known attribute-based cryptosystem from "learning with errors", a (conjectured) quantum-resistant hardness assumption tied to many lattice problems.

Acknowledgments

The author would like to thank Dan Boneh for suggesting a simplification of the scheme and its proof by way of the $\mathsf{ExtendRight}$ abstraction, and to thank the TCC 2013 program committee for what appears to be a very thorough review.

References

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology—EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, 2010.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *Advances in Cryptology—CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, 2010.

[3] Shweta Agrawal, Xavier Boyen, Vinod Vaikunthanathan, Panagiotis Voulgaris, and Howteck Wee. Functional encryption for threshold functions (or, fuzzy ibe) from lattices. In *Public Key Cryptography—PKC 2012*, 2012.

[4] Shweta Agrawal, David Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT 2011*, 2011.

[5] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC 1996*, pages 99–108. ACM, 1996.

[6] Miklos Ajtai. Generating hard instances of the short basis problem. In *ICALP*, volume 1644 of *LNCS*, pages 1–9. Springer, 1999.

[7] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.

[8] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS*, pages 75–86, 2009.

[9] A. Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Department of Computer Science, Technion, 1996.

[10] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS 2007*, pages 647–657, 2007.

[11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[12] Xavier Boyen. Lattice mixing and vanishing trapdoors – a framework for fully secure short signatures and more. In *Public Key Cryptography—PKC 2010*, volume 6056 of *LNCS*, pages 499–517, 2010.

[13] Xavier Boyen. Attribute-based functional encryption on lattices. Cryptology ePrint Archive, Report 2012/716, 2012. `http://eprint.iacr.org/`.

[14] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO 2011*, 2011.

[15] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees or, how to delegate a lattice basis. In *EUROCRYPT 2010*, 2010.

[16] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 26–8, 2001.

[17] Microsoft Corporation. Attribute based encryption using lattices. US patent application US20120155635, 17 December 2010.

[18] Leo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In *ASIACRYPT 2012*, LNCS. Springer, 2012.

[19] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In *STOC '08 – Proc. 40th ACM Symposium on the Theory of Computing*. ACM, 2008.

[20] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[21] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, pages 116–137, 2010.

[22] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008.

[23] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS 2006*, pages 89–98. ACM, 2006.

[24] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, pages 146–162, 2008.

[25] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351, 2010. `http://eprint.iacr.org/`.

[26] Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT 2012*, 2012.

[27] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *FOCS*, pages 356–365, 2002.

[28] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *STOC 2010*, pages 351–358. ACM, 2010.

[29] Victor Miller. The Weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4), 2004.

[30] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC 2009*, pages 333–342. ACM, 2009.

[31] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM J. Computing*, 40(6):1803–44, 2011.

[32] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.

[33] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93. ACM, 2005.

[34] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[35] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

[36] Brent Waters. Functional encryption for regular languages. In *CRYPTO 2012*, 2012.

A Background

In this Appendix we provide some abbreviated background information on lattice definitions and a few important results to be used in this paper. This also serves to fix the notation.

A.1 Random Integer Lattices

Throughout the paper, we let the parameters $q = q(\lambda), m = m(\lambda)$, $n = n(\lambda)$ are polynomial functions of the security parameter λ.

Definition 7. Let $\mathbf{B} = \left[\mathbf{b}_1 \,\middle|\, \ldots \,\middle|\, \mathbf{b}_m \right] \in \mathbb{R}^{m \times m}$ be an $m \times m$ matrix whose columns are linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^m$. The m-dimensional full-rank lattice Λ generated by $\mathbf{B}$ is the infinite periodic set,

$$\Lambda = \mathcal{L}(\mathbf{B}) = \left\{ \mathbf{y} \in \mathbb{R}^m \quad \text{s.t.} \quad \exists \mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{Z}^m \ , \ \ \mathbf{y} = \mathbf{B}\,\mathbf{s} = \sum_{i=1}^{m} s_i\, \mathbf{b}_i \right\}$$

Here, we are interested in integer lattices, i.e, infinite periodic subsets of $\mathbb{Z}^m$, that are invariant under translation by multiples of some integer q in each of the coordinates.

Definition 8. For q prime and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{u} \in \mathbb{Z}_q^n$, define:

$$\begin{aligned} \Lambda_q^{\perp}(\mathbf{A}) &= \left\{ \mathbf{e} \in \mathbb{Z}^m \quad \text{s.t.} \quad \mathbf{A}\,\mathbf{e} = 0 \pmod q \right\} \\ \Lambda_q^{\mathbf{u}}(\mathbf{A}) &= \left\{ \mathbf{e} \in \mathbb{Z}^m \quad \text{s.t.} \quad \mathbf{A}\,\mathbf{e} = \mathbf{u} \pmod q \right\} \end{aligned}$$

A.2 Discrete Gaussians

Definition 9. Let $m \in \mathbb{Z}_{>0}$ be a positive integer and $\Lambda \subset \mathbb{R}^m$ an m-dimensional lattice. For any vector $\mathbf{c} \in \mathbb{R}^m$ and any positive parameter $\sigma \in \mathbb{R}_{>0}$, we define:

$\rho_{\sigma,c}(\mathbf{x}) = \exp\left(-\pi \frac{\|\mathbf{x}-\mathbf{c}\|^2}{\sigma^2}\right)$: a Gaussian-shaped function on $\mathbb{R}^m$ with center $\mathbf{c}$ and parameter σ,

$\rho_{\sigma,\mathbf{c}}(\Lambda) = \sum_{\mathbf{x} \in \Lambda} \rho_{\sigma,\mathbf{c}}(\mathbf{x})$: the (always converging) discrete integral of $\rho_{\sigma,\mathbf{c}}$ over the lattice Λ,

$\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$: the discrete Gaussian distribution over Λ with center $\mathbf{c}$ and parameter σ,

$$\forall \mathbf{y} \in \Lambda \quad , \quad \mathcal{D}_{\Lambda,\sigma,\mathbf{c}}(\mathbf{y}) = \frac{\rho_{\sigma,\mathbf{c}}(\mathbf{y})}{\rho_{\sigma,\mathbf{c}}(\Lambda)}$$

For notational convenience, $\rho_{\sigma,0}$ and $\mathcal{D}_{\Lambda,\sigma,0}$ are abbreviated as ρ_σ and $\mathcal{D}_{\Lambda,\sigma}$.

A.3 Trapdoors for Lattices

Ajtai [6] showed how to sample an essentially uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with an associated full-rank set $\mathbf{T}_\mathbf{A} \subset \Lambda^{\perp}(\mathbf{A})$ of *low-norm vectors*. We will use an improved version of Ajtai's basis sampling algorithm due to Alwen and Peikert [8]:

Proposition 10 ([8])**.** Let $n = n(\lambda), q = q(\lambda), m = m(\lambda)$ be positive integers with $q \geq 2$ and $m \geq 5n \log q$. There exists a probabilistic polynomial-time algorithm $\mathsf{TrapGen}$ that outputs a pair $(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{T}_\mathbf{A} \in \mathbb{Z}_q^{m \times m})$, such that $\mathbf{A}$ is statistically close to uniform and $\mathbf{T}_\mathbf{A}$ is a basis for $\Lambda^{\perp}(\mathbf{A})$ with length $L = \widetilde{\|\mathbf{T}_\mathbf{A}\|} \leq m \cdot \omega(\sqrt{\log m})$ with all but $n^{-\omega(1)}$ probability.

TrapGen(n, m, q, σ) **[22]:** On input a modulus q, a lattice dimension m, a constraint dimension n, and a Gaussian deviation parameter σ dimension Λ, it outputs $\mathbf{A}$ and $\mathbf{T}$ as above.

A.3.1 Sampling Discrete Gaussians over Lattices

Gentry, Peikert and Vaikuntanathan [22] construct the following algorithm for sampling from the discrete Gaussian $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$, given a basis $\mathbf{B}$ for the m-dimensional lattice Λ with $\sigma \geq \|\widetilde{\mathbf{B}}\| \cdot \omega(\sqrt{\log m})$:

Proposition 11 ([22])**.** There exists a probabilistic polynomial-time algorithm that, on input an arbitrary basis $\mathbf{B}$ of an m-dimensional full-rank lattice $\Lambda = \mathcal{L}(\mathbf{B})$, a parameter $\sigma \geq \|\widetilde{B}\| \cdot \omega(\sqrt{\log m})$, and a center $\mathbf{c} \in \mathbb{R}^m$, outputs a sample from a distribution that is statistically close to $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$. For concreteness, we will refer to the algorithm of Proposition 11 as follows:

SampleGaussian$(\Lambda, \mathbf{B}, \sigma, \mathbf{c})$ **[22]:** On input lattice Λ, a basis $\mathbf{B}$ for Λ, a positive Gaussian parameter σ, and a center vector $\mathbf{c} \in \mathbb{R}^m$, it outputs a fresh random vector $\mathbf{x} \in \Lambda$ drawn from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$.

A.4 Preimage Sampling

The main use of short lattice basis for our purposes, is that they will allow us to sample short preimages of a specific target under the linear map defined by the matrix associated with the lattice. The following algorithm from [22] is what allows us to perform this preimage sampling. The shorter the lattice basis, the smaller a preimage we shall be able to obtain.

SamplePreimage$(\mathbf{A}, \mathbf{T_A}, \mathbf{u}, \sigma)$**:** Let $q \geq 2$, $m \geq 2n \log q$. On input a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ with 'short' trapdoor basis $\mathbf{T_A}$ for $\Lambda_q^{\perp}(\mathbf{A})$, a target image $\mathbf{u} \in \mathbb{Z}_q^n$ and a Gaussian parameter $\sigma \geq \|\widetilde{\mathbf{T_A}}\| \cdot \omega(\sqrt{\log m})$, outputs a sample $\mathbf{e} \in \mathbb{Z}^m$ from a distribution that is within negligible statistical distance of $\mathcal{D}_{\Lambda_q^{\mathbf{u}}(\mathbf{A}),\sigma}$.

A.5 Hardness Assumption

The LWE (learning with errors) problem was first defined by [33], and has since been extensively studied and used. We use the decisional version of the LWE problem.

Definition 12. Consider a prime q, a positive integer n, and a distribution χ over $\mathbb{Z}_q$, all public. An $(\mathbb{Z}_q, n, \chi)$-LWE problem instance consists of access to an unspecified challenge oracle $\mathcal{O}$, being, either, a noisy pseudo-random sampler $\mathcal{O}_s$ carrying some constant random secret key $\mathbf{s} \in \mathbb{Z}_q^n$, or, a truly random sampler $\mathcal{O}_\$$, whose behaviors are respectively as follows:

$\mathcal{O}_s$**:** outputs noisy pseudo-random samples of the form $(\mathbf{w}_i, v_i) = \left(\mathbf{w}_i,\ \mathbf{w}_i^T\,\mathbf{s} + x_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where, $\mathbf{s} \in \mathbb{Z}_q^n$ is a uniformly distributed persistent secret key that is invariant across invocations, $x_i \in \mathbb{Z}_q$ is a freshly generated ephemeral additive noise component with distribution χ, and $\mathbf{w}_i \in \mathbb{Z}_q^n$ is a fresh uniformly distributed vector revealed as part of the output.

$\mathcal{O}_\$$**:** outputs truly random samples $\left(\mathbf{w}_i, v_i\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, drawn independently uniformly at random in the entire domain $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The $(\mathbb{Z}_q, n, \chi)$-LWE problem statement, or LWE for short, allows an unspecified number of queries to be made to the challenge oracle $\mathcal{O}$, with no stated prior bound. We say that an algorithm $\mathcal{A}$ decides the $(\mathbb{Z}_q, n, \chi)$-LWE problem if $\left|\Pr[\mathcal{A}^{\mathcal{O}_s} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_\$} = 1]\right|$ is non-negligible for a random $s \in \mathbb{Z}_q^n$.

It has been shown in [33] that there is a $poly(n, q)$-time reduction from Search LWE$(\mathbb{Z}_q, n, \chi)$ to Decision LWE$(\mathbb{Z}_q, n, \chi)$.

The confidence in the hardness of the LWE problem stems in part from a result of Regev [33] which shows that the for certain noise distributions χ, the LWE problem is as hard as the worst-case SIVP and GapSVP under a quantum reduction (see also [30]). A classical reduction with related parameters was later obtained by Peikert [30].

Proposition 13 ([33])**.** Consider a real parameter $\alpha = \alpha(n) \in (0, 1)$ and a prime $q = q(n) > 2\sqrt{n}/\alpha$. Denote by $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ the group of reals $[0, 1)$ with addition modulo 1. Denote by Ψ_α the distribution over $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\,\pi}$ then reduced modulo 1. Denote by $\lfloor x \rceil = \lfloor x + \frac{1}{2} \rfloor$ the nearest integer to the real $x \in \mathbb{R}$. Denote by $\bar{\Psi}_\alpha$ the discrete distribution over $\mathbb{Z}_q$ of the random variable $\lfloor q\,X \rceil \mod q$ where the random variable $X \in \mathbb{T}$ has distribution Ψ_α.

Then, if there exists an efficient, possibly quantum, algorithm for deciding the $(\mathbb{Z}_q, n, \bar{\Psi}_\alpha)$-LWE problem, there exists a quantum $q \cdot poly(n)$-time algorithm for approximating the SIVP and GapSVP problems, to within $\tilde{O}(n/\alpha)$ factors in the ℓ_2 norm, in the worst case.

Since the best known algorithms for 2^k-approximations of gapSVP and SIVP run in time $2^{\widetilde{O}(n/k)}$) [19, 35, 28], it follows from the above that the LWE problem with the noise ratio $\alpha = 2^{-n^\epsilon}$ is likely hard for some constant $\epsilon < 1$.

B Example

Nothing like a small example could possibly illustrate better the workings of our KP-ABE scheme, and, by generalization, the fundamental principle behind the new framework that we propose for this kind of constructions.

Extraction. Consider the monotone access structure given by the Boolean expression,

$$\mathsf{Policy} \;\equiv\; (A \wedge B) \vee (A \wedge C)$$

Of course Policy can be reduced into the simpler expression $A \wedge (B \vee C)$, but the above expression has a duplicate attribute which will let us illustrate how that works. To construct the corresponding private key, we first rewrite the policy in function of uniquely appearing augmented attributes:

$$\mathsf{Policy}' \;\equiv\; (A^{(1)} \wedge B^{(2)}) \vee (A^{(3)} \wedge C^{(4)})$$

where the superscripts specify the arbitrary indices of the augmented attributes that together form the domain of the map ρ. We then translate Policy' into a sharing matrix, using some simple deterministic translation rule that for $\vee$-&-$\wedge$ formulas guarantees a binary output:

$$\mathsf{L}_{(1\wedge 2)\vee(3\wedge 4)} \;=\; \begin{matrix}(1)\\(2)\\(3)\\(4)\end{matrix}\!\!\begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 0 & 1\end{bmatrix}$$

where indeed for $d \neq 0$ the reconstruction target vector $[d \quad 0 \quad 0]$ is in the span of exactly those subsets of the rows of L whose corresponding augmented attributes satisfy our Policy'. Starting from L we assemble the "virtual encryption matrix" M which here takes the form,

$$\mathsf{M} \;=\; \left[\begin{array}{cccc|c|cc} \mathsf{A}_A & & & & \mathsf{A}_0 & \mathsf{Z}_1 & \\ & \mathsf{A}_B & & & & \mathsf{Z}_1 & \\ & & \mathsf{A}_A & & \mathsf{A}_0 & & \mathsf{Z}_2 \\ & & & \mathsf{A}_C & & & \mathsf{Z}_2 \end{array}\right] \bmod q$$

where we observe that the public matrix A_A appears twice on the left diagonal, because there were two occurrences of the attribute A in the access structure formula Policy. From there, knowing a trapdoor for each of the A_i on the left diagonal, we can construct our policy-encoding private key K as a (randomized) short basis for the lattice induced by M. The private key hides the structure of M and especially the ephemeral and undisclosed submatrices Z_i from the recipient.

$$\mathsf{K} \;=\; \left[\begin{array}{ccccc|c} & & & & & \\ & & & & & \\ & & & & & \\ & & \mathsf{K}' & & & \\ \hline & & & & & \\ & & & & & \end{array}\right]$$

As mentioned before, only the upper-left quadrant K' of K is useful; the rest may be omitted.

Encryption. To encrypt, say, for the set of attributes $\mathsf{Attrib} = \{A, C\}$, or in other words for the binary attribute vector $\mathsf{Attrib} = (1, 0, 1) \in \{0,1\}^{\{A,B,C\}}$, the encryptor first constructs the Regev encryption matrix,

$$\mathsf{F} \;=\; \left[\begin{array}{cccc|c} \mathsf{A}_A & 0 & \mathsf{A}_C & & \mathsf{A}_0 \end{array}\right]$$

and then proceeds to create from it a randomized Regev ciphertext,

$$\mathsf{Ctx} \;=\; \Big(\mathrm{c}_0 = \mathbf{s}^\top \mathbf{u} + \lfloor \frac{q}{2} \rfloor \, \mathsf{Msg} + \nu_0,\; \mathbf{c}_1 = \mathbf{s}^\top \mathsf{F} + \nu_1 \Big) \bmod q$$

where for the sequel we note that $\mathbf{c}_1$ decomposes into four subvectors: one for each of the three original attributes (notice how B is actually missing) plus one for the submatrix A_0, written,

$$\mathbf{c}_1 \;=\; \left[\begin{array}{cccc} \mathbf{c}_{1,A} & \mathbf{0} & \mathbf{c}_{1,C} & \mathbf{c}_{1,0} \end{array}\right]$$

We emphasize that since encryption is with reference to a list of attributes, there is no notion of duplication here.

Decryption. To decrypt Ctx using Key, the first step is to find a linear combination of the rows of L that allows reconstruction of the target vector. Here, the winning combination is a ternary[5] vector $\mathbf{g}$, where,

$$\underbrace{\left[\begin{array}{cccc} g_{A1} & g_B & g_{A2} & g_C \\ 0 & 0 & 1 & -1 \end{array}\right]}_{\mathbf{g}} \cdot \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \;=\; \underbrace{\left[\begin{array}{ccc} d & & \\ 1 & 0 & 0 \end{array}\right]}_{\text{target}}$$

[5] As long as there are only $\vee$ and $\wedge$ but no threshold gates, we can choose whether the sharing matrix has binary and the reconstruction vector ternary coefficients in $\mathbb{Z}$, or *vice versa*. This choice does not affect the objects' norms.

where we note not only that $\mathbf{g}$ assigns non-zero coefficients exclusively to augmented attributes corresponding to the attributes A and C that are "true" in the ciphertext, but also that, of the two augmented attributes $A^{(1)}$ and $A^{(3)}$ linked to A, one has value 1 and the other 0. This is consistent with the rules of secret-sharing reconstruction with duplicated attributes, which require only that all "false" attributes be given coefficient zero. Next, per the Decrypt algorithm, we construct a "real encryption matrix" M' using the coefficients from $\mathbf{g}$ and d,

$$\mathsf{M}' \;=\; \left[\begin{array}{cccc|c|cc} 0 & 0 & \mathsf{A}_A & -\mathsf{A}_C & \mathsf{A}_0 & 0 & 0 \end{array}\right]$$

and we know that this matrix M' can be expressed from the "virtual encryption matrix" M (for which K is a short basis and K' an excerpt of same) by the "tensor" linear combination,

$$(\mathbf{g} \otimes \boxed{1}) \;\cdot\; \mathsf{M} \;=\; \mathsf{M}'$$

The Decrypt algorithm shows how to transform the key K' into a trapdoor for the matrix M' (or precisely for the matrix F'' reconstructed from M' to match the true encryption matrix F). Here, the transformation is as follows:

$$\mathsf{K}' \;=\; \left[\begin{array}{ccc|c} & \mathbf{k}_j^{(1)} & & \\ & \mathbf{k}_j^{(2)} & & \\ \cdots & \mathbf{k}_j^{(3)} & \cdots & \cdots \\ & \mathbf{k}_j^{(4)} & & \\ \hline & \mathbf{k}_j^{(0)} & & \end{array}\right] \qquad \text{begets} \qquad \mathsf{T}'' \;=\; \left[\begin{array}{ccc} & \mathbf{0} & \\ & \mathbf{0} & \\ \cdots & \mathbf{k}_j^{(3)} & \cdots \\ & -\mathbf{k}_j^{(4)} & \\ & \mathbf{k}_j^{(0)} & \end{array}\right]$$

The last step, in order to perform Regev decryption using a short preimage $\mathbf{d}$ of $\mathbf{u}$ sampled using this trapdoor T'', is to have a matching ciphertext, *i.e.*, a ciphertext (that appears to have been) built upon the "real" encryption matrix M' (or, rather, its pruned version, the aforementioned F''). Since some attributes were duplicated in the matrix M' (or F''), we need to perform a similar duplication of the corresponding ciphertext fragments, and transform,

$$\mathbf{c}_1 \;=\; [\,\mathbf{c}_{1,A} \quad \mathbf{0} \quad \mathbf{c}_{1,C} \quad \mathbf{c}_{1,0}\,] \qquad \text{into} \qquad \mathbf{c}'_1 \;=\; [\,\mathbf{c}_{1,A} \quad \mathbf{0} \quad \mathbf{c}_{1,A} \quad \mathbf{c}_{1,C} \quad \mathbf{c}_{1,0}\,]$$

so that the width of $\mathbf{c}'_1$ matches that of M' to ensure that Regev decryption can proceed. Decryption proper works first by using the trapdoor T'' to sample a preimage $\mathbf{d}$ of $\mathbf{u}$ under M' (or F''), and then by taking the inner product of said $\mathbf{d}$ with $\mathbf{c}'_1$ and subtracting the result (consisting of blinding plus noise) from $\mathbf{c}_0$ to reveal (a noisy version of) $\lfloor \frac{q}{2} \rfloor\, \mathsf{Msg} \pmod q$.

Remark. The formal description of the scheme provides that the superfluous rows and columns of F'', T'', $\mathbf{c}'_1$, *etc.*—*i.e.*, such rows and columns that become multiplied by zero—, be suppressed for efficiency. We preserved them in this illustration for the sake of exposition.

LSSS. To fix ideas, it is also useful to see a few other examples of explicit encodings of policies into sharing matrices. The following examples illustrate, with one exception, cases of formulas that consists only of $\vee$-gates and $\wedge$-gates with no non-trivial threshold gates. In that case, regardless of input duplication, it is always possible to have binary sharing matrices, *i.e.*, with entries in $\{0,1\}$, that allow ternary reconstruction vectors, *i.e.*, with coefficients in $\{0,\pm 1\}$.

$$\mathsf{L}_{\vee(1,2)} \;=\; \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \mathsf{L}_{\geq 2(1,2,3)} \;=\; \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

$$\mathsf{L}_{\wedge(1,2,3)} \;=\; \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathsf{L}_{\wedge(1,\vee(2,\wedge(3,4),\wedge(5,6)))} \;=\; \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$