ETSI TR 103 618 V1.1.1 (2019-12)

CYBER;
Quantum-Safe Identity-Based Encryption

Reference
DTR/CYBER-QSC-0012

Keywords
encryption, identity, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document describes a proposal for a quantum-safe hierarchical identity-based encryption scheme. It gives an overview of the functionality provided by hierarchical identity-based encryption, outlines some example uses cases and provides a high-level description of a potential solution based on structured lattices. The description includes concrete proposals for parameter sets, estimates for performance in software and a practical security analysis.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] A. Shamir: "Identity-based cryptosystems and signature schemes", CRYPTO, 1984.

[i.2] J. Bethencourt, A. Sahai and B. Waters: "Ciphertext-Policy Attribute-Based Encryption", Security and Privacy, 2007.

[i.3] C. Gentry and A. Silverberg: "Hierarchical ID-Based Cryptography", ASIACRYPT, 2001.

[i.4] D. Boneh and M. Franklin: "Identity-Based Encryption from the Weil Pairing", CRYPTO, 2001.

[i.5] A. Boldyreva, V. Goyal and V. Kumar: "Identity-based Encryption with Efficient Revocation", CCS, 2008.

[i.6] J. H. Seo and K. Emura: "Revocable Identity-Based Encryption Revisited: Security Model and Construction", PKC, 2013.

[i.7] X. Ding and G. Tsudik: "Simple Identity-Based Cryptography with Mediated RSA", CT-RSA, 2003.

[i.8] K. Paterson and G. Price: "A comparison between traditional public key infrastructures and identity-based cryptography", Information Security Technical Report 8(3), 57-72, 2003.

[i.9] P. Szczechowiak and M. Collier: "TinyIBE: Identity-based encryption for heterogeneous sensor networks", Intelligent Sensors, Sensor Networks and Information Processing, 2009.

[i.10] ETSI EN 300 392-7: "Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 7: Security".

[i.11] ETSI EN 300 396-6: "Terrestrial Trunked Radio (TETRA); Direct Mode Operation (DMO); Part 6: Security".

[i.12] SAFEcrypto: "D9.1 - Case study specifications and requirements", June 2015.

NOTE: Available at https://www.safecrypto.eu/outcomes/deliverables.

[i.13] C. Cocks: "An identity based encryption scheme based on quadratic residues", IMA International Conference on Cryptography and Coding, 2001.

[i.14] C. Gentry, C. Peikert and V. Vaikuntanathan: "How to Use a Short Basis: Trapdoors for Hard Lattices and New Cryptographic Constructions", STOC, 2008.

[i.15] D. Cash, D. Hofheinz, E. Kiltz, C. Peikert: "Bonsai trees, or how to delegate a lattice basis", J. Cryptology 25(4), 601-639, 2012.

[i.16] S. Agrawal, D. Boneh and X. Boyen: "Efficient lattice (H)IBE in the standard model", EUROCRYPT, 2010.

[i.17] S. Agrawal, D. Boneh and X. Boyen: "Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE", CRYPTO, 2010.

[i.18] L. Ducas, V. Lyubashevsky and T. Prest: "Efficient identity-based encryption over NTRU lattices", ASIACRYPT, 2014.

[i.19] P. Bert, P.-A. Fouque, A. Roux-Langlois and M. Sabt: "Practical implementation of Ring-SIS/LWE based signature and IBE", Post-Quantum Cryptography, 2018.

[i.20] S. McCarthy, N. Smyth and E. O'Sullivan: "A practical implementation of identity-based encryption over NTRU lattices", IMA International Conference on Cryptography and Coding, 2017.

[i.21] T. Güneysu and T. Oder: "Towards lightweight identity-based encryption for the post-quantum-secure Internet of Things", Quality Electronic Design, 2017.

[i.22] P. Klein: "Finding the closest lattice vector when it's unusually close", SODA, 2000.

[i.23] P. Q. Nguyen and O. Regev: "Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures", EUROCRYPT, 2006.

[i.24] D. Micciancio and S. Goldwasser: "Complexity of lattice problems: A cryptographic perspective", Kluwer Academic Publishers, Boston, 2002.

[i.25] V. Lyubashevsky, C. Peikert and O. Regev: "A Toolkit for Ring-LWE Cryptography", EUROCRYPT, 2013.

[i.26] P. Campbell and M. Groves: "Practical post-quantum Hierarchical Identity-Based Encryption", IMA Conference on Cryptography and Coding, 2017.

[i.27] S. Fluhrer: "Cryptanalysis of Ring-LWE based key exchange with key share reuse", IACR ePrint Archive 2016/085, 2016.

[i.28] E. Fujisaki and T. Okamoto: "Secure integration of asymmetric and symmetric encryption schemes", CRYPTO, 1999.

[i.29] T. Pöppelmann and T. Güneysu: "Towards practical lattice-based public-key encryption on reconfigurable hardware", SAC, 2013.

[i.30] M. Abe, R. Gennaro, K. Kurosawa and V. Shoup: "Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM", EUROCRYPT, 2005.

[i.31] E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, T. Pöppelmann, P. Schwabe and D. Stebila: "NewHope: Algorithm specifications and supporting documentation", NIST First Round Post-Quantum Submission, 2017.

[i.32] V. Lyubashevsky and T. Prest: "Quadratic time, linear space algorithms for Gram-Schmidt orthogonalization and Gaussian sampling in structured lattices", EUROCRYPT, 2015.

[i.33] SAFEcrypto: "WP6: libsafecrypto".

NOTE: Available at https://www.github.com/safecrypto/libsafecrypto.

[i.34] T. Pornin and T. Prest: "More efficient algorithms for the NTRU key generation using the field norm", PKC, 2019.

[i.35] L. Ducas and T. Prest: "Fast Fourier orthogonalization", ISSAC, 2016.

[i.36] D. Stebila and M. Mosca: "Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project", SAC, 2016.

NOTE: Available at https://www.github.com/open-quantum-safe/liboqs.

[i.37] M. Albrecht, F. Göpfert, F. Virdia and T. Wunderer: "Revisiting the expected cost of solving uSVP and applications to LWE", ASIACRYPT, 2017.

[i.38] A. Becker, L. Ducas, N. Gama and T. Laarhoven: "New directions in nearest neighbor searching with applications to lattice sieving", SODA, 2016.

[i.39] M. Albrecht, Y. Lindell, E. Orsini, V. Osheter, K. Paterson, G. Peer and N. Smart: "LIMA: A PQC encryption scheme", NIST First Round Post-Quantum Submission, 2017.

[i.40] T. Laarhoven: "Search problems in cryptography: From fingerprinting to lattice sieving", PhD thesis, Eindhoven University of Technology, 2015.

[i.41] C. Peikert: "How (not) to instantiate Ring-LWE", SCN, 2016.

[i.42] V. Lyubashevsky C. Peikert and O. Regev: "On ideal lattices and learning with errors over rings", EUROCRYPT, 2010.

[i.43] M.-J. Saarinen: "Ring-LWE ciphertext compression and error correction: Tools of lightweight post-quantum cryptography", IoTPTS, 2017.

[i.44] P. Longa and M. Naehrig: "Speeding up the Number Theoretic Transform for faster ideal lattice-based cryptography", CANS, 2016.

[i.45] C. Peikert: "Lattice cryptography for the internet", Post-Quantum Cryptography, 2014.

[i.46] J.-P. D'Anvers, F. Vercauteren and I. Verbauwhede: "On the impact of decryption failures on the security of LWE/LWR based schemes", IACR ePrint Archive 2018/1089, 2018.

[i.47] E. Alkim, L. Ducas, T. Pöppelmann and P. Schwabe: "Post-quantum key exchange - a new hope", USENIX Security, 2016.

[i.48] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte and Z. Zhang: "FALCON: Fast-Fourier lattice-based compact signatures over NTRU", NIST First Round Post-Quantum Submission, 2017.

[i.49] P. Kirchner and P.-A. Fouque: "Revisiting lattice attacks on overstretched NTRU parameters", EUROCRYPT, 2017.

[i.50] J. Buchmann, F. Göpfert, R. Player and T. Wunderer: "On the Hardness of LWE with Binary Error: Revisiting the Hybrid Lattice-Reduction and Meet-in-the-Middle Attack", AFRICACRYPT, 2016.

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

$\bar{a}$	Transpose of the polynomial a
(a)	Vector of coefficients of the polynomial a
⌊a⌉	Co-ordinatewise rounding of the polynomial a
‖a‖	Euclidean norm of the vector a
$a \cdot b$	Multiplication of the polynomials a and b
$a * b$	Co-ordinatewise multiplication of the vectors a and b
$a \mid\mid b$	Concatenation of the strings a and b
$a \oplus b$	Exclusive or of the values a and b
$\mathrm{Adv}(\mathcal{A})$	Advantage of the adversary $\mathcal{A}$
$\mathcal{B}^*$	Gram-Schmidt vectors corresponding to the basis $\mathcal{B}$
$\|\mathcal{B}\|_{\mathrm{GS}}$	Gram-Schmidt norm of the basis $\mathcal{B}$
$D(\mu, \sigma)$	Discrete Gaussian distribution with mean μ and standard deviation σ
Γ	Gamma function
$\mathcal{M}(a)$	Matrix representation of the polynomial a
$\mathbb{Q}$	Rational numbers
$\mathbb{R}$	Real numbers
$\mathrm{Res}(a, b)$	Resultant of the polynomials a and b
$\mathbb{Z}$	Integers

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ABB	Agrawal, Boneh and Boyen
ABE	Attribute-Based Encryption
AMD	Advanced Micro Devices
AVX	Advanced Vector eXtensions
BKZ	Block Korkine-Zolotarev
CA	Certificate Authority
CCA	Chosen-Ciphertext Attack
CPA	Chosen-Plaintext Attack
CRL	Certificate Revocation List
DLP	Ducas, Lyubashevsky and Prest
GPV	Gentry, Peikert and Vaikuntanathan
HIBE	Hierarchical Identity-Based Encryption
IBE	Identity-Based Encryption
IND	INDistinguishability
IP	Internet Protocol
KDF	Key Derivation Function
KEM	Key Encapsulation Mechanism
KMS	Key Management Service
LWE	Learning With Errors
NIST	National Institute of Standards and Technology
NTT	Number-Theoretic Transform
OCSP	Online Certificate Status Protocol
PKI	Public-Key Infrastructure
QSC	Quantum-Safe Cryptography
SEM	SEcurity Mediator
TETRA	TErrestrial Trunked RAdio
URL	Universal Resource Locator

4 Identity-Based Encryption (IBE)

4.1 Introduction

In public-key cryptography each user has a key pair consisting of matched public and private keys.

Traditionally, the private key is generated first via a random process and the public key is derived from the private key via a mathematical function that is hard to invert. Public keys constructed in this way are pseudo-random and have no intrinsic meaning. Consequently, it is usually necessary to bind the public key to a public identifier associated to the user; e.g. the user's e-mail address, their device's Internet Protocol (IP) address or their website's Universal Resource Locator (URL). The binding is typically achieved by including the public key and identifier in a certificate that is digitally signed by a trusted third party such as a Certification Authority (CA) during a certification process.

In contrast, with identity-based cryptography [i.1] the public key is chosen first and the private key is derived from the public key. This means that a user's public key can have some intrinsic semantic value. Specifically, it can be chosen to be the representation of a public identifier associated with the user. The most important difference between traditional public-key cryptography and identity-based cryptography is that the user's private key needs to be derived from their identifier by a trusted third party such as a Key Management Service (KMS) during a registration process.

More generally, the public keys can include auxiliary information the user such as their employment status, authorizations or geographical location. This allows finer-grained access control as the KMS can verify that the user holds the appropriate authorizations before issuing the corresponding private key. A more flexible version of this functionality is provided by attribute-based cryptography [i.2] where, for example, data can be protected in such a way that only users whose attributes satisfy a certain policy are able to access it.

4.2 Functionality

One of the main advantages of identity-based cryptography is that it offers the possibility of lightweight key management without the need for certificates or a full public-key infrastructure (PKI).

If Alice wants to send Bob a message protected by a public-key encryption scheme where the public keys are managed by a PKI, then she first needs to obtain the certificate containing Bob's public key either directly from Bob or from a central certificate repository (Figure 1).

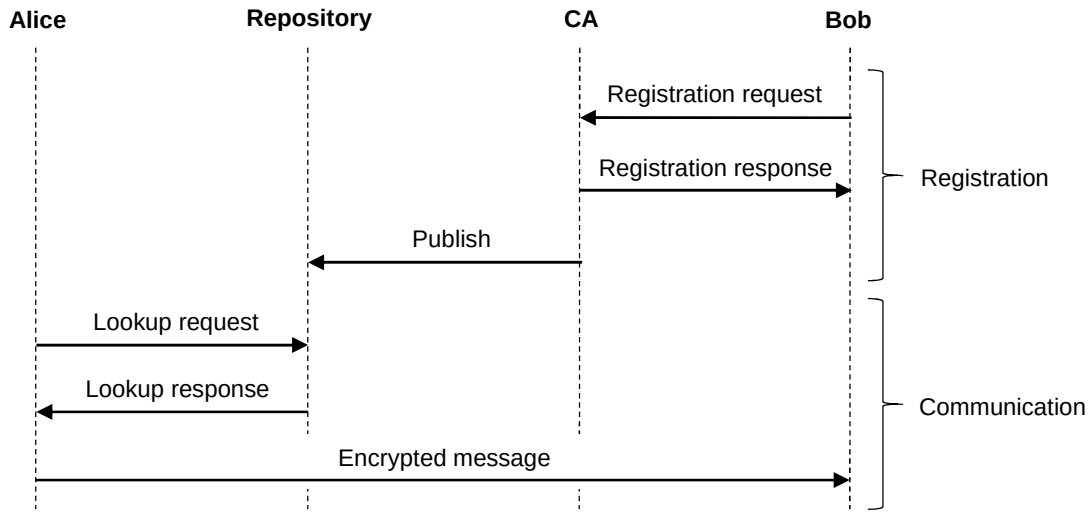

Figure 1: Encrypted communication with a PKI

If Alice wants to send Bob a message protected by an identity-based encryption (IBE) scheme, then she only needs to know Bob's public identifier as this corresponds to his authenticated public key. This can enable simplex transmission of encrypted messages without the involvement of the KMS (Figure 2). It is even possible for Alice to send Bob an encrypted message before he has registered and been given his private key.

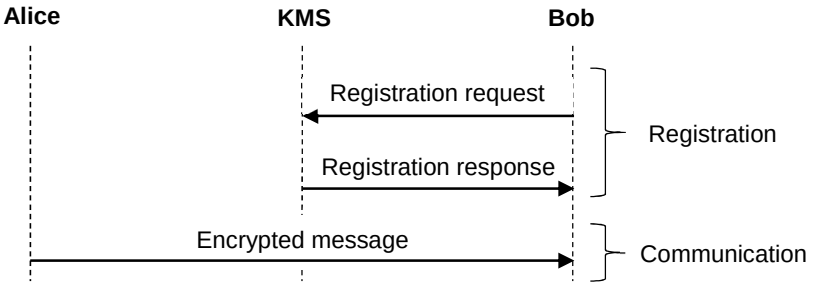

Figure 2: Encrypted communication with IBE

PKIs that handle a large number of users typically involve multiple levels of CAs. For example, in a two-tier model the root CA delegates authority to one or more issuing CAs who then sign the certificates containing user public keys. Hierarchical identity-based encryption (HIBE) [i.3] is an analogous concept. A central KMS delegates the ability to derive user private keys to one or more a sub-KMSs. This provides more scalable and flexible user management, and still allows simplex transmission of encrypted messages without the involvement of a KMS (Figure 3).

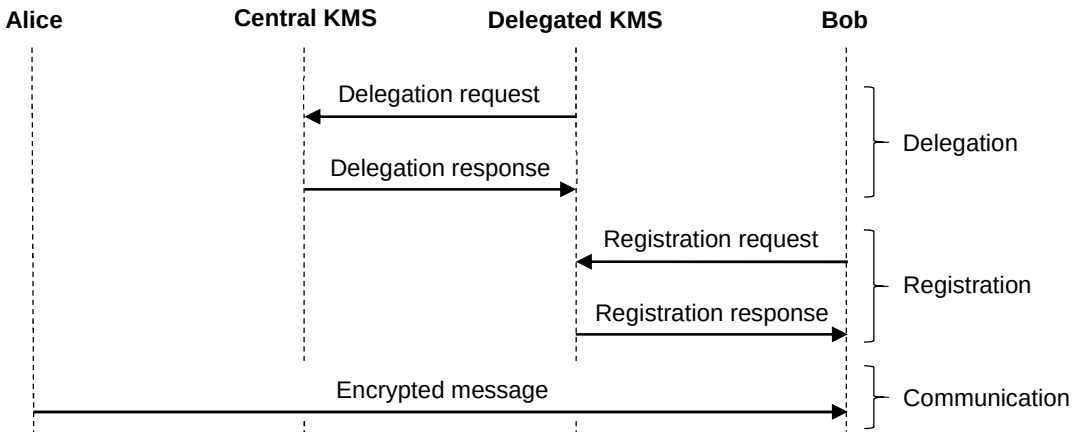

Figure 3: Encrypted communication with HIBE

In practice, traditional public-key cryptography is often used in an authenticated key exchange to establish a shared symmetric key between two or more users. Identity-based cryptography can be used to provide similar functionality. In this case, Alice generates a symmetric key and sends it to Bob encrypted under his public identifier. If Bob can successfully decrypt the symmetric key, then this implicitly authenticates Bob to Alice. For mutual authentication, Alice can use an identity-based signature to digitally sign the message with a key that is bound to her public identifier. Alternatively, Bob can send Alice a response message that is encrypted under her public identifier (Figure 4).

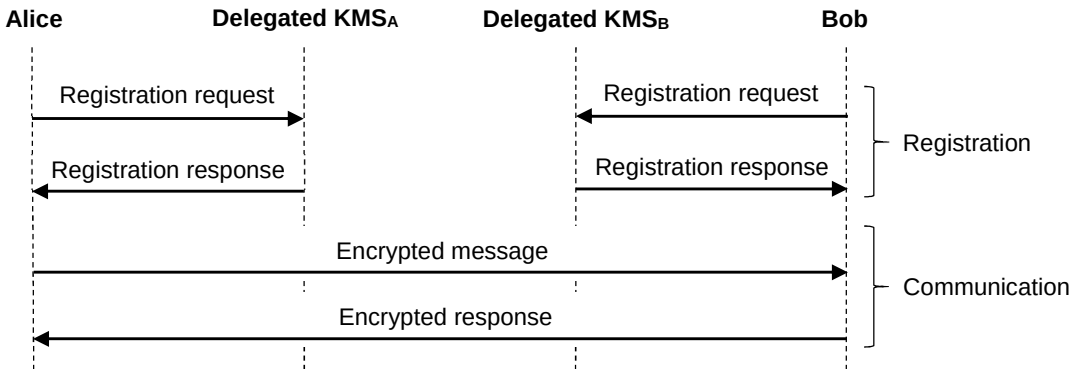

Figure 4: Mutual authentication with HIBE

4.3 Discussion

A fundamental feature of most IBE and attribute-based encryption (ABE) schemes is the reliance on a trusted KMS to derive user private keys based on their public identifiers or attributes. In a traditional PKI, if an adversary can gain access to the CA or compromise its private key, then they are potentially able to impersonate any user in the system and read encrypted communications via man-in-the-middle attacks. However, if an adversary can gain access to the KMS or compromise its private key, then they are potentially able to read any encrypted communications on the system including messages that were sent before the comprise. There are several responses and mitigations to this:

- To be able to read a user's communications an attacker would need both to obtain the private key and be able to intercept or otherwise access the encrypted messages. In many real-wold deployments, the KMS is based in a secure location and user key derivation is performed off-line. Network access is only required during the user provisioning process itself which is typically only performed at initial registration and then potentially at monthly or yearly intervals after that.

- The use of HIBE can further limit the exposure of the central KMS as network access is only required during the provisioning of a small number of sub-KMSs which is likely to be infrequent. Similarly, the compromise of a sub-KMS only affects the users managed by that KMS and KMSs below it in the hierarchy rather than all users in the system.

- There are cryptographic mechanisms that allow split or multi-party derivation of the user private keys with a distributed KMS [i.4] that requires the co-operation of more than one authority. The shares of the user private key can then be stored at different secure locations. An adversary would need to gain access to multiple authorities or comprise private data in multiple places in order to recover the full private key for a user.

- For some deployments, there are valid requirements for the organization to be able to access user private keys. For example, there might be regulatory requirements to audit encrypted communications on the enterprise network. Similarly, it allows the recovery of encrypted corporate data in the event that a user loses their private key. In examples such as these, it is important that policies are put in place to ensure that access is restricted to properly authorized individuals for valid regulatory or organizational reasons, and that they are only allowed access to a limited set of well-specified private keys.

The other area where IBE schemes differ significantly from a traditional PKI is revocation of compromised user private keys. In a PKI, the revocation is typically handled using Certificate Revocation Lists (CRLs) periodically issued by the CA, or checks performed via the Online Certificate Status Protocol (OCSP). However, revocation is more complicated for IBE schemes as a user's private key is intrinsically linked to their identifier. There are a few different approaches that can be taken:

- The simplest option is to include a time and date as part of the public key in order to limit the validity period for the compromised private key [i.4]. The equivalent of a CRL could then be used to prevent further messages being encrypted to the compromised user for the remainder of the validity period. Messages encrypted before the compromise would still potentially be vulnerable. Further, all users in the system would need to securely contact the KMS to obtain their new private keys for next validity period.

- More efficient versions of revocable IBE have been proposed [i.5], [i.6]. These separate the identifier and timestamp components of the user private key to allow public key updates from the KMS. They also include a binary tree structure to reduce the cost of revoking the private key for a specific user. Messages encrypted during the validity period for a compromised private key would still be vulnerable.

- An alternative option is to use mediated IBE [i.7] where a user is given a share of their private key and can only decrypt messages with the help of a trusted Security Mediator (SEM). Following a compromise of the user's private key share, the SEM could then prevent the decryption of any messages until the user had been issued a new private key share. However, the SEM would need to be involved in every encrypted communication.

For a more detailed discussion of the differences between identity-based cryptography and traditional PKI see [i.8].

4.4 Example use cases

IBE and ABE are particularly appropriate for government or enterprise applications where registration of employees onto a private enterprise network could be handled by the human resources department and be subject to policy checks on the employee. Further, with HIBE the headquarters of a large organization could retain control of the central KMS while allowing local provisioning of users by sub-KMSs situated in regional offices. At the opposite end of the spectrum, IBE also seems suitable for some Internet of Things applications where keys can be generated centrally by the KMS and distributed to low-power embedded devices [i.9].

A third set of use cases is around public safety and mission critical applications which require secure one-to-one, group and broadcast communications. Low-latency key management is important so that a secure communication channel is available as soon as possible during an incident and there is a need to support direct communications between users when the network infrastructure is unavailable. Existing public-safety networks such as TETRA [i.10], [i.11] typically rely on pre-placed symmetric keys, but IBE and ABE have been considered for group management by the SAFEcrypto project [i.12].

SAFEcrypto [i.12] have also considering ABE for privacy-preserving analysis of municipal data so that it can be securely used by researchers investigating topics such as public health, economic and social trends, and crime prevention. Different levels of access might be needed for individuals in different roles; for example, municipality staff, academic researchers, and administrators from the cloud service provider hosting the data.

4.5 Quantum-safe IBE

Although identity-based cryptography was originally proposed by Shamir [i.1] in 1994, the first IBE schemes did not appear until 2001 when the pairing-based scheme by Boneh and Franklin [i.4] and the quadratic-residue-based scheme by Cocks [i.13] were published. Most IBE schemes that have been proposed since then are similarly based on pairings on elliptic curves. However, elliptic curve cryptography is not quantum-safe, so a symmetric key encrypted with IBE today will be vulnerable to a future adversary with access to a large-scale quantum computer.

Several quantum-safe IBE and HIBE schemes have been proposed in the academic literature [i.14], [i.15], [i.16] and [i.17] and the most practical of these use structured lattices [i.18], [i.19]. Encryption and decryption is reported to be several times faster than comparable pairing-based schemes [i.20], [i.21], although ciphertexts can be significantly larger.

The present document gives a high-level description of a HIBE scheme based on structured lattices (clause 5), with concrete parameter suggestions (clause 6) and performance estimates (clause 7). A discussion of the security of the scheme is provided in annex C.

5 Lattice-based HIBE

5.1 Background

The first lattice-based IBE scheme was proposed by Gentry, Peikert and Vaikuntanathan (GPV) [i.14]. It was based on a Learning with Errors (LWE) encryption scheme where the lattice was constructed to have a known short basis. User key extraction was performed using a modification of the randomized nearest-plane algorithm by Klein [i.22]. This approach ensured that the user private keys did not leak information about the private basis for the lattice, preventing statistical attacks such as [i.23].

The GPV scheme was later adapted to give a HIBE scheme based on LWE by Cash et al [i.15]. The hierarchical construction used a "bonsai tree" approach where the lattice for the central KMS was extended to a higher-dimensional lattice for the sub-KMS. The central KMS was able to delegate a short basis for the extended lattice by combining the modified Klein sampling technique [i.14] and a method for randomizing a basis for a lattice from [i.24].

A slightly different HIBE approach was proposed by Agrawal, Boneh and Boyen (ABB) [i.17]. This used the same delegation technique as [i.15], but the lattice for the sub-KMS was a fixed-dimensional modification of the lattice for the central KMS rather than a higher-dimensional extension.

Although the "toolkit" paper [i.42] described a Ring-LWE encryption scheme that would be suitable for IBE, the first full description of an IBE scheme based on Ring-LWE was given by Ducas, Lyubashevsky and Prest (DLP) [i.18]. It modified the GPV scheme [i.14] to use a lattice with an NTRU-style trapdoor and the Ring-LWE encryption scheme from [i.42]. More recently, Bert et al [i.19] have proposed a Ring-LWE version of the ABB standard model IBE scheme from [i.16].

The present document describes LATTE [i.26], a Ring-LWE based HIBE scheme that adapts the DLP IBE scheme [i.18] so that it works with the bonsai tree construction of Cash et al [i.15]. An overview of the two-level version of LATTE is given in clause 5.2. Details of the general scheme are given in clauses 5.3 onwards.

5.2 Overview

5.2.1 Polynomial ring

LATTE uses a polynomial ring of the form

$$R = \mathbb{Z}[x]/(x^n + 1)$$

for a power-of-two n, and its quotient

$$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$

for a prime q.

A polynomial in R can be reduced to a polynomial in R_q by reducing each coefficient modulo q. A polynomial in R_q can be lifted to a polynomial in R by lifting each coefficient to an integer in the range $\{-(q-1)/2, \ldots, (q-1)/2\}$.

A polynomial $p = p_0 + p_1 x + \cdots + p_{n-1} x^{n-1} \in R$ can be represented by an n-long vector of coefficients

$$(p) = (p_0, p_1, p_2, \ldots, p_{n-1})$$

or an $n \times n$ matrix of integers

$$\mathcal{M}(p) = \begin{bmatrix} p_0 & p_1 & p_2 & \cdots & p_{n-1} \\ -p_{n-1} & p_0 & p_1 & \cdots & p_{n-2} \\ -p_{n-2} & -p_{n-1} & p_0 & \cdots & p_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -p_1 & -p_2 & -p_3 & \cdots & p_0 \end{bmatrix}$$

where the i-th row corresponds to the vector of coefficients of the polynomial $x^{i-1} \cdot p \in R$.

5.2.2 Central KMS lattice

The lattice $\mathcal{L}_0$ for the central KMS is constructed with an NTRU-style trapdoor. The public basis for $\mathcal{L}_0$ has the form

$$\mathcal{B}_0 = \begin{bmatrix} q\,I_n & 0_n \\ \mathcal{M}(A) & I_n \end{bmatrix}$$

where I_n is the $n \times n$ identity matrix, 0_n is the $n \times n$ zero matrix, and A is a polynomial in R_q.

The master public key for the central KMS is the pair (A, B) where B is an auxiliary polynomial chosen uniformly at random from R_q.

The private basis for the lattice $\mathcal{L}_0$ has the form

$$\mathcal{S}_0 = \begin{bmatrix} \mathcal{M}(g) & \mathcal{M}(f) \\ \mathcal{M}(G) & \mathcal{M}(F) \end{bmatrix}$$

where f, g, G and F are small polynomials in R. In particular, $A \in R_q$ is constructed as

$$A \equiv g \cdot f^{-1} \pmod{q}$$

and the private basis needs to satisfy the 2-dimensional determinant condition

$$g \cdot F - f \cdot G = q$$

over R. The master private key for the central KMS is the 2×2 matrix

$$S_0 = \begin{bmatrix} g & f \\ G & F \end{bmatrix}$$

over R corresponding to the private basis $\mathcal{S}_0$ of $\mathcal{L}_0$.

5.2.3 Sub-KMS lattice

The lattice $\mathcal{L}_1$ for a sub-KMS with identifier ID_1 corresponds to a higher-dimensional extension of the lattice $\mathcal{L}_0$ for the central KMS. The public basis for the extended lattice $\mathcal{L}_1$ is

$$\mathcal{B}_1 = \begin{bmatrix} q\,I_n & 0_n & 0_n \\ \mathcal{M}(A_0) & I_n & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n \end{bmatrix}$$

where $A_0 = A$ is the first component of the master public key and $A_1 = H(ID_1)$ is a polynomial in R_q obtained from the identifier ID_1 for the sub-KMS via a hash function

$$H: \{0,1\}^* \to R_q.$$

The delegated private basis for the extended lattice $\mathcal{L}_1$ has the form

$$\mathcal{S}_1 = \begin{bmatrix} \mathcal{M}(s_{0,0}) & \mathcal{M}(s_{0,1}) & \mathcal{M}(s_{0,2}) \\ \mathcal{M}(s_{1,0}) & \mathcal{M}(s_{1,1}) & \mathcal{M}(s_{1,2}) \\ \mathcal{M}(s_{2,0}) & \mathcal{M}(s_{2,1}) & \mathcal{M}(s_{2,2}) \end{bmatrix}$$

where the $s_{i,j}$ are small polynomials in R. The private key for the sub-KMS is derived by the KMS using the master private key. In particular, it needs to satisfy the 3-dimensional determinant condition

$$\det \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} \\ s_{1,0} & s_{1,1} & s_{1,2} \\ s_{2,0} & s_{2,1} & s_{2,2} \end{bmatrix} = q$$

over R. The delegated private key for the sub-KMS is the 3×3 matrix

$$S_1 = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} \\ s_{1,0} & s_{1,1} & s_{1,2} \\ s_{2,0} & s_{2,1} & s_{2,2} \end{bmatrix}$$

over R corresponding to the short basis $\mathcal{S}_1$ of the extended lattice $\mathcal{L}_1$.

5.2.4 User lattice

The lattice $\mathcal{L}_2$ for a user with identifier ID_2 and managed by a sub-KMS with identifier ID_1 corresponds to a higher-dimensional extension of the lattice $\mathcal{L}_1$ for the sub-KMS. The public basis for the user's lattice $\mathcal{L}_2$ is

$$\mathcal{B}_2 = \begin{bmatrix} q\,I_n & 0_n & 0_n & 0_n \\ \mathcal{M}(A_0) & I_n & 0_n & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n & 0_n \\ \mathcal{M}(A_2) & 0_n & 0_n & I_n \end{bmatrix}$$

where $A_2 = H(ID_1 \,||\, ID_2)$ is a polynomial in R_q obtained from the hash of the identifiers ID_1 and ID_2 for the delegated KMS and user.

The private key for the user is a quadruple (t_0, t_1, t_2, t_3) of small polynomials in R such that

$$A_0 \cdot t_0 + A_1 \cdot t_1 + A_2 \cdot t_2 + t_3 \equiv B \pmod{q}$$

over R_q. The user private key is derived by the delegated KMS using their delegated private key.

5.2.5 Encryption

A message is encoded as a polynomial $m \in R$ with coefficients from $\{0, (q-1)/2\}$. It is encrypted by sampling small polynomials $e, e_0, e_1, e_2, e_3 \in R$ and forming

$$\begin{aligned} C_0 &\equiv A_0 \cdot e + e_0 && \pmod{q} \\ C_1 &\equiv A_1 \cdot e + e_1 && \pmod{q} \\ C_2 &\equiv A_2 \cdot e + e_2 && \pmod{q} \\ C_3 &\equiv B \cdot e + e_3 + m && \pmod{q}. \end{aligned}$$

The ciphertext is the quadruple (C_0, C_1, C_2, C_3) of polynomials in R_q.

The ciphertext is decrypted by forming

$$\begin{aligned} V &\equiv C_3 - C_0 \cdot t_0 - C_1 \cdot t_1 - C_2 \cdot t_2 && \pmod{q} \\ &\equiv e_3 - e_0 \cdot t_0 - e_1 \cdot t_1 - e_2 \cdot t_2 + e \cdot t_3 + m && \pmod{q}, \end{aligned}$$

lifting to R, and then rounding to recover m.

5.2.6 Validation

In a HIBE scheme, a user's public key is determined by their identifier and so is static. This means that lattice-based HIBE schemes will be vulnerable to active attacks that use malformed ciphertexts to provoke decryption failures [i.27]. Consequently, LATTE includes a variant of the Fujisaki-Okamoto transform [i.28] to allow validation of ciphertexts.

5.3 Parameters

The main public parameters for LATTE are:

- n, a power of two defining the dimension for the cyclotomic ring $R = \mathbb{Z}[x]/(x^n + 1)$;
- q, a prime defining the modulus for the quotient ring $R_q = R/qR$; and
- d, the maximum number of levels in the hierarchy.

The modulus q is chosen with $q \equiv 1 \pmod{2n}$ so that the Number-Theoretic Transform (NTT) can be used for efficient arithmetic in the ring R_q (see clause B.2).

The parameters n and q determine the parameters for the discrete Gaussian distributions used in master key generation, delegation and extraction (see clause 6.1):

- σ_0, the standard deviation for the master private key; and
- σ_ℓ, the standard deviation for a delegated private key or user private key in level ℓ.

Additional parameters for encryption and decryption are:

- σ_e, the standard deviation for the ephemeral private keys; and
- u, the number of coefficients used to encode a message bit (see clause 5.7).

NOTE: Key generation, delegation and extraction all use discrete Gaussians to prevent statistical key recovery attacks (see clause C.2.1). However, these are not relevant to encryption so for this it is enough to use a distribution that is approximately Gaussian such as a zero-centred binomial distribution.

LATTE also uses a hash function $H : \{0,1\}^* \to R_q$ to convert identifiers into uniformly random polynomials in the ring and a Key Derivation Function $\mathrm{KDF} : \{0,1\}^* \to \{0,1\}^{256}$ to produce 256-bit values for the encryption process.

5.4 Key generation

The main part of key generation for the central KMS is the construction of an NTRU lattice $\mathcal{L}_0$ with a private basis of the form

$$S_0 = \begin{bmatrix} \mathcal{M}(g) & \mathcal{M}(f) \\ \mathcal{M}(G) & \mathcal{M}(F) \end{bmatrix}$$

for small polynomials $f, g, G, F \in R$. However, the Gram-Schmidt norm of S_0 (see clause A.1.2) determines the size of the private keys that can safely be used in the next level of the hierarchy (see clause 6.1) so key generation needs to include a check that it is sufficiently small. This check can be performed efficiently since, by Lemma 3 of [i.18],

$$\|S_0\|_{\mathrm{GS}} = \mathrm{Max}\left(\|(g,f)\|, \left\|\left(\frac{-q\bar{f}}{f\cdot\bar{f}+g\cdot\bar{g}}, \frac{q\bar{g}}{f\cdot\bar{f}+g\cdot\bar{g}}\right)\right\|\right)$$

where for $p = p_0 + p_1 x + \cdots + p_{n-1}x^{n-1} \in R$ the transposed polynomial is defined as $\bar{p} = p_0 - p_{n-1}x - \cdots - p_1 x^{n-1}$.

Key generation for LATTE closely follows the approach in [i.18]:

1) The KMS generates polynomials $f, g \in R$ by sampling their coefficients independently from a discrete Gaussian $D(0, \sigma_0)$ with mean 0 and standard deviation σ_0.

2) The KMS computes the Gram-Schmidt norm of the private basis corresponding to f and g by computing

$$N = \mathrm{Max}\left(\|(g,f)\|, \left\|\left(\frac{-q\bar{f}}{f\cdot\bar{f}+g\cdot\bar{g}}, \frac{q\bar{g}}{f\cdot\bar{f}+g\cdot\bar{g}}\right)\right\|\right)$$

If $N > \sqrt{2n}\cdot\sigma_0$, then the KMS returns to step 1.

3) The KMS uses the extended Euclidean algorithm to find the resultant $r_f = \mathrm{Res}(f, x^n+1) \in \mathbb{Z}$ and the corresponding polynomial $u_f \in R$ such that

$$f \cdot u_f = r_f$$

over R. If $r_f \equiv 0 \pmod q$ then the KMS returns to step 1.

4) The KMS uses the extended Euclidean algorithm to find the resultant $r_g = \mathrm{Res}(g, x^n+1) \in \mathbb{Z}$ and the corresponding polynomial $u_g \in R$ such that

$$g \cdot u_g = r_g$$

over R.

5) The KMS uses the extended Euclidean algorithm to find $v_f, v_g \in \mathbb{Z}$ such that

$$v_f r_f + v_g r_g = 1$$

over $\mathbb{Z}$. If $\gcd(r_f, r_g) > 1$, then the KMS returns to step 1.

6) The KMS computes the polynomials $F' = q v_g \cdot u_g$ and $G' = -q v_f \cdot u_f$ in R.

At this stage, the polynomials F' and G' satisfy the relation

$$\begin{aligned} g \cdot F' - f \cdot G' &= (q v_g \cdot u_g) \cdot g + (q v_f \cdot u_f) \cdot f \\ &= q v_g r_g + q v_f r_f \\ &= q, \end{aligned}$$

over R, but the coefficients of F' and G' are multiples of q. The KMS can reduce them using f and g.

7) The KMS computes the polynomials $F = F' - \lfloor k \rceil \cdot f$ and $G = G' - \lfloor k \rceil \cdot g$ in R where

$$k = \frac{F' \cdot \bar{f} + G' \cdot \bar{g}}{f \cdot \bar{f} + g \cdot \bar{g}}$$

in $\mathbb{Q}[x]/(x^n + 1)$.

8) The KMS computes the master public key (A, B) where A is given by

$$A \equiv g \cdot f^{-1} \pmod{q}$$

and B is sampled uniformly at random from R_q.

9) The KMS returns the master public key (A, B) and the master private key

$$S_0 = \begin{bmatrix} g & f \\ G & F \end{bmatrix}.$$

5.5 Delegation

Let ID_ℓ be the identifier for a sub-KMS at level ℓ that is managed by an $(\ell - 1)$-long chain of KMSs with identifiers $ID_1, \ldots, ID_{\ell-1}$.

The delegated private basis for the sub-KMS at level ℓ is a short basis

$$\mathcal{S}_\ell = \begin{bmatrix} \mathcal{M}(s_{0,0}) & \mathcal{M}(s_{0,1}) & \cdots & \mathcal{M}(s_{0,\ell+1}) \\ \mathcal{M}(s_{1,0}) & \mathcal{M}(s_{1,1}) & & \mathcal{M}(s_{1,\ell+1}) \\ \vdots & & \ddots & \\ \mathcal{M}(s_{\ell+1,0}) & \mathcal{M}(s_{\ell+1,1}) & & \mathcal{M}(s_{\ell+1,\ell+1}) \end{bmatrix}$$

for the lattice $\mathcal{L}_\ell$ with public basis

$$\mathcal{B}_\ell = \begin{bmatrix} q I_n & 0_n & 0_n & \cdots & 0_n \\ \mathcal{M}(A_0) & I_n & 0_n & & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n & & 0_n \\ \vdots & & & \ddots & \\ \mathcal{M}(A_\ell) & 0_n & 0_n & & I_n \end{bmatrix}$$

where $A_0 = A$ is the first component of the master public key and $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$.

The vector $(s_{i,0}, \ldots, s_{i,\ell+1})$ is a short vector in the lattice $\mathcal{L}_\ell$ precisely when the polynomials $s_{i,0}, \ldots, s_{i,\ell+1} \in R$ form a short solution to the equation

$$A_0 \cdot s_{i,1} + A_1 \cdot s_{i,2} + \cdots + A_\ell \cdot s_{i,\ell+1} \equiv s_{i,0} \pmod{q}$$

over R_q.

Further, the matrix $\mathcal{S}_\ell$ of short vectors in $\mathcal{L}_\ell$ will give a basis for the lattice precisely when the polynomials $s_{i,j} \in R$ satisfy the condition

$$\det \begin{bmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,\ell+1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,\ell+1} \\ \vdots & \vdots & & \vdots \\ s_{\ell,0} & s_{\ell,1} & \cdots & s_{\ell,\ell+1} \\ s_{\ell+1,0} & s_{\ell+1,1} & \cdots & s_{\ell+1,\ell+1} \end{bmatrix} = q$$

over R.

The KMS at level $\ell - 1$ uses these two observations to construct a delegated private basis for the sub-KMS at level ℓ as follows:

1) The KMS computes the hashes $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$.

2) For each $i \in \{0, \ldots, \ell\}$:

   2.1) The KMS generates a polynomial $s_{i,\ell+1} \in R$ by sampling its coefficients independently from a discrete Gaussian $D(0, \sigma_\ell)$ with mean 0 and standard deviation σ_ℓ.

   2.2) The KMS uses the Klein sampler for the lattice $\mathcal{L}_{\ell-1}$ given by

$$\mathcal{B}_{\ell-1} = \begin{bmatrix} q\,I_n & 0_n & 0_n & \cdots & 0_n \\ \mathcal{M}(A_0) & I_n & 0_n & & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n & & 0_n \\ \vdots & & & \ddots & \\ \mathcal{M}(A_{\ell-1}) & 0_n & 0_n & & I_n \end{bmatrix}$$

   with the KMS's private basis $\mathcal{S}_{\ell-1}$, centre vector

$$c = (-A_\ell \cdot s_{i,\ell+1}, 0, 0, \ldots, 0)$$

   and target standard deviation σ_ℓ, to obtain a lattice vector

$$v = \left(s'_{i,0}, s'_{i,1}, s'_{i,2}, \ldots, s'_{i,\ell}\right).$$

   2.3) The KMS sets $s_{i,0} = s'_{i,0} + A_\ell \cdot s_{i,\ell+1}$ and $s_{i,j} = s'_{i,j}$ for $j \in \{1, \ldots, \ell\}$.

   2.4) If $\left\|(s_{i,0}, \ldots, s_{i,\ell+1})\right\| > \sqrt{(\ell+2)n} \cdot \sigma_\ell$, then the KMS returns to step 2.1.

The Klein sampler in step 2.2 is used to give polynomials $s'_{i,0}, s'_{i,1}, s'_{i,2}, \ldots, s'_{i,\ell} \in R$ that satisfy the equation

$$A_0 \cdot s_{i,1} + A_1 \cdot s_{i,2} + \cdots + A_{\ell-1} \cdot s_{i,\ell} \equiv s_{i,0} - A_\ell \cdot s_{i,\ell+1} \pmod{q}$$

over R, but appear to have been sampled from a discrete Gaussian $D(0, \sigma_\ell)$ with mean 0 and standard deviation σ_ℓ.

The remaining polynomials $s_{\ell+1,0}, \ldots, s_{\ell+1,\ell+1} \in R$ are constructed using a higher-dimensional analogue of steps 3 to 6 from key generation.

3) For each $j \in \{0, \ldots, \ell + 1\}$:

   3.1) The KMS computes the cofactor

$$M_j = (-1)^{j+\ell+1} \det \begin{bmatrix} s_{0,0} & \cdots & s_{0,j-1} & s_{0,j+1} & \cdots & s_{0,\ell+1} \\ s_{1,0} & \cdots & s_{1,j-1} & s_{1,j+1} & \cdots & s_{1,\ell+1} \\ \vdots & & \vdots & \vdots & & \vdots \\ s_{\ell,0} & \cdots & s_{\ell,j-1} & s_{\ell,j+1} & \cdots & s_{\ell,\ell+1} \end{bmatrix}$$

   over R.

3.2) The KMS uses the extended Euclidean algorithm to find the resultant $r_j = \mathrm{Res}(M_j, x^n + 1) \in \mathbb{Z}$ and the corresponding polynomial $u_j \in R$ such that

$$M_j \cdot u_j = r_j$$

over R.

4) The KMS uses a sequence of extended Euclidean algorithms to find $v_0, \ldots, v_{\ell+1} \in \mathbb{Z}$ such that

$$v_0 r_0 + \cdots + v_{\ell+1} r_{\ell+1} = 1$$

over $\mathbb{Z}$. If this is not possible, then the KMS returns to step 2.

5) The KMS computes the polynomials $s_{\ell+1,j} = q v_j \cdot u_j \in R$ for $j \in \{0, \ldots, \ell+1\}$.

At this stage

$$\det \begin{bmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,\ell+1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,\ell+1} \\ \vdots & \vdots & & \vdots \\ s_{\ell,0} & s_{\ell,1} & \cdots & s_{\ell,\ell+1} \\ s_{\ell+1,0} & s_{\ell+1,1} & \cdots & s_{\ell+1,\ell+1} \end{bmatrix} = q$$

over R, but the entries $s_{\ell+1,j}$ in the final row have coefficients that are multiples of q. These can be reduced using a generalization of step 7 from key generation.

6) The KMS computes

$$c = \det \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,\ell} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,\ell} \\ \vdots & \vdots & & \vdots \\ a_{\ell,0} & a_{\ell,1} & \cdots & a_{\ell,\ell} \end{bmatrix}$$

over R, where $a_{i,j} = s_{j,0} \cdot \bar{s}_{i,0} + \cdots + s_{j,\ell+1} \cdot \bar{s}_{i,\ell+1}$ for $i, j \in \{0, \ldots, \ell\}$.

7) For each $j \in \{0, \ldots, \ell\}$:

7.1) The KMS computes

$$d_j = \det \begin{bmatrix} a_{0,0} & \cdots & a_{0,j-1} & b_0 & a_{0,j+1} & \cdots & a_{0,\ell} \\ a_{1,0} & \cdots & a_{1,j-1} & b_1 & a_{1,J+1} & \cdots & a_{1,\ell} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{\ell,0} & \cdots & a_{\ell,j-1} & b_\ell & a_{\ell,j+1} & \cdots & a_{\ell,\ell} \end{bmatrix}$$

over R, where $b_i = s_{\ell+1,0} \cdot \bar{s}_{i,0} + \cdots + s_{\ell+1,\ell+1} \cdot \bar{s}_{i,\ell+1}$ for $i \in \{0, \ldots, \ell\}$.

8) The KMS sets

$$(s_{\ell+1,0}, \ldots, s_{\ell+1,\ell+1}) = (s_{\ell+1,0}, \ldots, s_{\ell+1,\ell+1}) - \lfloor k_0 \rceil \cdot (s_{0,0}, \ldots, s_{0,\ell+1}) - \cdots - \lfloor k_\ell \rceil \cdot (s_{\ell,0}, \ldots, s_{\ell,\ell+1})$$

where $k_j = d_j / c$ in $\mathbb{Q}[x]/(x^n + 1)$.

9) The KMS returns the delegated private key

$$S_\ell = \begin{bmatrix} s_{0,0} & s_{0,1} & \cdots & s_{0,\ell+1} \\ s_{1,0} & s_{1,1} & \cdots & s_{1,\ell+1} \\ \vdots & \vdots & & \vdots \\ s_{\ell,0} & s_{\ell,1} & \cdots & s_{\ell,\ell+1} \\ s_{\ell+1,0} & s_{\ell+1,1} & \cdots & s_{\ell+1,\ell+1} \end{bmatrix}$$

5.6 Extraction

Let ID_ℓ be the identifier of a user at level ℓ managed by an $(\ell - 1)$-long chain of KMSs with identifiers $ID_1, \ldots, ID_{\ell-1}$.

The private key for the user is a short solution $t_0, \ldots, t_{\ell+1} \in R$ to the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \equiv B \pmod{q}$$

over R_q, where $A_0 = A$ is the first component of the master public key, $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$, and B is the second component of the master public key.

The KMS at level $\ell - 1$ extracts a private key for the user at level ℓ as follows:

1) The KMS computes the hashes $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$.

2) The KMS generates a small polynomial $t_\ell \in R$ by sampling its coefficients independently from a discrete Gaussian $D(0, \sigma_\ell)$ with mean 0 and standard deviation σ_ℓ.

3) The KMS uses the Klein sampler for the lattice $\mathcal{L}_{\ell-1}$ given by

$$\mathcal{B}_{\ell-1} = \begin{bmatrix} q\,I_n & 0_n & 0_n & \cdots & 0_n \\ \mathcal{M}(A_0) & I_n & 0_n & & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n & & 0_n \\ \vdots & & & \ddots & \\ \mathcal{M}(A_{\ell-1}) & 0_n & 0_n & & I_n \end{bmatrix}$$

with the KMS's private basis $\mathcal{S}_{\ell-1}$, centre vector

$$c = (B - A_\ell \cdot t_\ell, 0, 0, \ldots, 0)$$

and target standard deviation σ_ℓ, to obtain a lattice vector

$$v = (t'_0, t'_1, t'_2, \ldots, t'_{\ell-1}, t'_\ell).$$

4) The KMS sets $t_{\ell+1} = B - A_\ell \cdot t_\ell - t'_0$ and $t_j = t'_{j+1}$ for $j \in \{0, \ldots, \ell - 1\}$.

5) The KMS returns the user private key $(t_0, \ldots, t_{\ell+1})$.

The Klein sampler in step 3 is used to give polynomials $t_0, \ldots, t_{\ell-1}, t_{\ell+1} \in R$ that satisfy the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_{\ell-1} \cdot t_{\ell-1} + t_{\ell+1} \equiv B - A_\ell \cdot t_\ell \pmod{q}$$

over R_q, but appear to have been sampled from a discrete Gaussian $D(0, \sigma_\ell)$ with mean 0 and standard deviation σ_ℓ.

NOTE: The final component $t_{\ell+1}$ of the user private key is not needed for decryption, but it can be used to verify that the extracted user private key is valid by directly checking that the relation above holds.

5.7 Message encoding

Let $u = n/256$. Encryption and decryption use the u-to-1 threshold encoding scheme from [i.29]. A 256-bit message $\mu \in \{0,1\}^{256}$ is encoded as the polynomial $m \in R$ with coefficients in $\{0, (q-1)/2\}$ given by

$$m = \frac{q-1}{2} \sum_{i=0}^{255} \mu_i \left(x^{ui} + \cdots + x^{u(i+1)-1} \right).$$

A polynomial $m \in R$ with coefficients from $\{-(q-1)/2, \ldots, (q-1)/2\}$ is correspondingly decoded as a 256-bit message $\mu \in \{0,1\}^{256}$ where

$$\mu_i = \begin{cases} 0 & \text{if } |m_{ui}| + \cdots + \left|m_{u(i+1)-1}\right| < uq/4; \\ 1 & \text{otherwise.} \end{cases}$$

5.8 Encryption

Let ID_ℓ be the identifier of a user at level ℓ managed by an $(\ell - 1)$-long chain of KMSs with identifiers $ID_1, \ldots, ID_{\ell-1}$.

Encryption of a message follows the general Ring-LWE approach. The user encodes the message m, samples small polynomials $e, e_0, \dots, e_{\ell+1} \in R$ and forms

$$\begin{bmatrix} C_0 \\ \vdots \\ C_\ell \\ C_{\ell+1} \end{bmatrix} = \begin{bmatrix} A_0 \\ \vdots \\ A_\ell \\ B \end{bmatrix} \cdot e + \begin{bmatrix} e_0 \\ \vdots \\ e_\ell \\ e_{\ell+1} \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m \end{bmatrix} \pmod q$$

where $A_0 = A$ is the first component of the master public key, $A_i = H(ID_1 \,||\, \dots \,||\, ID_i)$ for $i \in \{1, \dots, \ell\}$ and B is the second component of the master public key.

Encryption also uses the variant of the Fujisaki-Okamoto transform from [i.30] to provide security against attacks that use malformed ciphertexts. A simple Key Derivation Function (KDF) is used to encrypt the message with a random seed value. The random seed is then encrypted using a Ring-LWE scheme where the ephemeral private keys are sampled deterministically from a discrete Gaussian.

A 256-bit message $\mu \in \{0,1\}^{256}$ is encrypted to the user as follows:

1) The KMS computes the hashes $A_i = H(ID_1 \,||\, \dots \,||\, ID_i)$ for $i \in \{1, \dots, \ell\}$.

2) The initiator generates a random 256-bit value seed and uses it to encrypt the message μ

$$Z = \mu \oplus \mathrm{KDF}(seed).$$

3) The initiator generates polynomials $e, e_0, \dots, e_{\ell+1} \in R$ by sampling their coefficients from an approximate discrete Gaussian $D(0, \sigma_e)$ with mean 0 and standard deviation σ_e using a deterministic process seeded by $\mathrm{KDF}(seed \,||\, Z)$.

4) The initiator forms the ephemeral public keys $C_0, \dots, C_\ell \in R_q$ by computing

$$C_i \equiv A_i \cdot e + e_i \pmod q,$$

where $A_0 = A$ is the first component of the master public key and $A_i = H(ID_1 \,||\, \dots \,||\, ID_i)$ for $i \in \{1, \dots, \ell\}$.

5) The initiator encodes $seed$ as the polynomial $m \in R$, as described in clause 5.7, and masks it by computing

$$C_{\ell+1} \equiv B \cdot e + e_{\ell+1} + m \pmod q,$$

where B is the second component of the master public key.

6) The initiator sends the user the ciphertext $(Z, C_0, \dots, C_\ell, C_{\ell+1})$.

5.9 Decryption

Let ID_ℓ be the identifier of a user at level ℓ managed by an $(\ell - 1)$-long chain of KMSs with identifiers $ID_1, \dots, ID_{\ell-1}$.

Decryption of a ciphertext $(C_0, \dots, C_{\ell+1})$ follows the general Ring-LWE approach. The user forms

$$\begin{aligned} V &\equiv C_{\ell+1} - C_0 \cdot t_0 - \dots - C_\ell \cdot t_\ell && \pmod q \\ &\equiv \frac{(q-1)}{2} m + e_{\ell+1} - e_0 \cdot t_0 - \dots - e_\ell \cdot t_\ell + e \cdot t_{\ell+1} && \pmod q, \end{aligned}$$

where $t_0, \dots, t_{\ell+1} \in R$ is the user's private key, and decodes the message m.

Decryption also includes the ciphertext validation approach from [i.30]. Ring-LWE decryption is used to recover the random seed from $(C_0, \dots, C_{\ell+1})$. The seed is then used to recreate the Ring-LWE encryption. If the reconstructed ciphertext $(C'_0, \dots, C'_{\ell+1})$ matches the received ciphertext $(C_0, \dots, C_{\ell+1})$, then the message is decrypted from Z. Otherwise, a decryption error is returned.

A ciphertext $(Z, C_0, \dots, C_\ell, C_{\ell+1})$ is decrypted as follows:

1) The user computes

$$V \equiv C_{\ell+1} - C_0 \cdot t_0 - \dots - C_\ell \cdot t_\ell \pmod q$$

over R_q, lifts it to a polynomial in R with coefficients in $\{-(q-1)/2, \ldots, (q-1)/2\}$, and decodes it to obtain a putative value $seed'$.

2) The user generates polynomials $e', e'_0, \ldots, e'_{\ell+1} \in R$ by sampling their coefficients from an approximate discrete Gaussian $D(0, \sigma_e)$ with mean 0 and standard deviation σ_e using a deterministic process seeded by $\mathrm{KDF}(seed' \,||\, Z)$.

3) The user forms the ephemeral public keys $C'_0, \ldots, C'_\ell \in R_q$ by computing

$$C'_i \equiv A_i \cdot e' + e'_i \pmod{q}$$

where $A_0 = A$ is the first component of the master public key and $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$.

4) The user encodes $seed'$ as the polynomial $m' \in R$, as described in clause 5.7, and masks it by computing

$$C'_{\ell+1} \equiv B \cdot e' + e'_{\ell+1} + m' \pmod{q},$$

where B is the second component of the master public key

5) If the values $(C'_0, \ldots, C'_\ell, C'_{\ell+1})$ reconstructed by the user match the values $(C_0, \ldots, C_\ell, C_{\ell+1})$ sent by the initiator, then the user returns the decrypted message

$$\mu' = Z \oplus \mathrm{KDF}(seed').$$

Otherwise, the user returns a decryption failure.

6 Parameter selection

6.1 Gaussian distributions

It is important to choose the standard deviations $\sigma_0, \ldots, \sigma_d$ for the discrete Gaussian distributions with care to ensure that the distribution of vectors produced by Klein sampling is close enough the target discrete Gaussian to prevent them leaking information about the private basis (see clause C.2.1).

The standard deviation for the master private keys is chosen to be

$$\sigma_0 \approx \sqrt{qe/4n}$$

to minimize the Gram-Schmidt norm of the master private basis $\mathcal{S}_0$ (see section 3 of [i.18]).

Similarly, the standard deviation for the delegated private keys and user private keys is chosen to be

$$\sigma_\ell \approx \eta'_\varepsilon(\mathbb{Z}) \cdot \sqrt{(\ell+1)n} \cdot \sigma_{\ell-1}$$

where $\eta'_\varepsilon(\mathbb{Z})$ is the smoothing parameter with an ε that allows 2^{48} samples to be produced by a single KMS without reducing the security of the scheme (see clause C.2.1).

Ephemeral private keys are assumed to have the fixed standard deviation

$$\sigma_e = 2{,}0.$$

This matches the New Hope NIST submission [i.31] where the coefficients of the private keys are sampled from a zero-centred binomial distribution with parameter $k = 8$.

6.2 Ring dimension and modulus

The various lattice attacks against LATTE are:

- Master key recovery as a short vector problem (see clause C.2.3);
- Delegated key recovery as a close vector problem (see clause C.2.4);

- User key recovery as a close vector problem (see clause C.2.5); and
- Message recovery as a close vector problem in the primal lattice or a short vector problem in the dual lattice (see clause C.2.6).

The choice of standard deviation for the master private key, delegated private key and user private key ensures that the cost of the first three attacks only depends on the dimension n of the cyclotomic ring. Consequently, the dimension is chosen to be large enough to block these attacks.

The standard deviation for the ephemeral private keys is fixed so, for a given ring dimension n, the cost of the message recovery problem increases as the modulus q decreases. However, LATTE is also vulnerable to active attacks that exploit naturally occurring decryption failures even when ciphertext validation is performed and the probability of a decryption failure decreases as the modulus increases (see clause C.2.2). Consequently, the modulus is chosen to balance the cost of the message recovery attack and the decryption failure attack.

6.3 Parameter sets

6.3.1 Single-level IBE scheme

Table 1 lists two potential parameter sets for LATTE as a single-level IBE scheme along with two parameter sets for DLP from the implementation paper [i.20].

Table 1: Parameter sets for single-level IBE schemes

Scheme	DLP-0	DLP-3	LATTE-1	LATTE-2
Security Target	80 bits	192 bits	128 bits	256 bits
n	512	1 024	1 024	2 048
q	$2^{22} + 2^{20} + 2^{19} + 1$	$2^{22} + 2^{20} + 2^{19} + 1$	$2^{24} - 2^{14} + 1$	$2^{25} - 2^{12} + 1$
d	1	1	1	1
σ_0	87,9 (note 1)	62,1 (note 1)	105,9	105,9
σ_1	3 184,5 (note 1)	3 225,0 (note 1)	5 499,6	7 880,6
σ_e	0,82 (note 2)	0,82 (note 2)	2,0	2,0
u	1	1	4	8
Master Public Key	1 472 bytes	2 944 bytes	6 144 bytes	12 800 bytes
Master Private Key	3 712 bytes	6 912 bytes	7 424 bytes	14 848 bytes
User Key	1 152 bytes	2 176 bytes	6 912 bytes	13 824 bytes
Ciphertext	1 792 bytes	3 712 bytes	9 248 bytes (note 3)	19 232 bytes (note 3)

NOTE 1: The DLP parameters in [i.20] do not include σ_0 or σ_1. The values for σ_0 and σ_1 given here are chosen using the approach described in clause 6.1.
NOTE 2: The coefficients of the ephemeral private keys in DLP are sampled uniformly from $\{-1,0,1\}$.
NOTE 3: The LATTE ciphertext sizes do not include the compression techniques discussed in clause B.1.

For comparison, the 256-bit secure parameters for the New Hope NIST submission [i.31] use ring dimension $n = 1\ 024$ and modulus $q = 2^{14} - 2^{12} + 1$ to give an 1 824-byte public key and a 2 208-byte ciphertext.

6.3.2 Two-level HIBE scheme

Table 2 lists potential parameter sets for LATTE as an HIBE scheme with two levels.

Table 2: Parameter sets for two-level HIBE schemes

Scheme	LATTE-3	LATTE-4
Security Target	~ 80 bits	160 bits
n	1 024	2 048
q	$2^{36} - 2^{20} + 1$	$2^{38} - 2^{26} + 1$
d	2	2
σ_0	6 777,4	9 583,5
σ_1	351 958,7	713 152,4
σ_2	22 559 368,5	65 487 839,3
σ_e	2,0	2,0
u	4	8
Master Public Key	9 216 bytes	19 456 bytes
Master Private Key	10 496 bytes	20 992 bytes
Delegated Key	29 568 bytes	61 440 bytes
User Key	15 360 bytes	31 744 bytes
Ciphertext	18 464 bytes (see note)	38 944 bytes (see note)
NOTE: The LATTE ciphertext sizes do not include the compression techniques discussed in clause B.1.		

6.3.3 Discussion

6.3.3.1 Master public key size

The master public key for LATTE contains a pair of elements from R_q so will be twice the size of the DLP master public key for similar parameters. It is possible to reduce its size by a half by considering the second component B to be a fixed system parameter rather than allowing the KMS to generate it.

6.3.3.2 Gram-Schmidt storage

The master private key for LATTE is a 2×2 matrix of small elements from R. The central KMS needs to compute the corresponding $2n$ Gram-Schmidt vectors to use in the Klein sampler for delegation and extraction. Representing the Gram-Schmidt vectors with 64-bit floating-point numbers takes 32 megabytes for both LATTE-1 and LATTE-3; and 128 megabytes for both LATTE-2 and LATTE-4.

Similarly, a delegated private key in two-level LATTE is a 3×3 matrix of small elements from R, but the sub-KMS needs to compute the corresponding $3n$ Gram-Schmidt vectors to use in the Klein sampler for extraction. Representing the vectors with 128-bit floating-point numbers takes 144 megabytes for LATTE-3 and 576 megabytes for LATTE-3.

A compact version of the sampler described by Lyubashevsky and Prest [i.32] recomputes the Gram-Schmidt vectors in reverse order during sampling using a relatively small amount of data saved from the initial Gram-Schmidt computation (see clause A.4). This reduces the Gram-Schmidt storage requirement for the central KMS by a factor of 512 for both LATTE-1 and LATTE-3; and a factor of 1 024 for both LATTE-2 and LATTE-4. The Gram-Schmidt storage requirement for the sub-KMS is reduced by a factor of 614 for LATTE-3 and a factor 1 229 for LATTE-4.

6.3.3.3 User private key size

A user private key for single-level LATTE contains a triple of small elements from R so will be three times the size of a DLP user private key for similar parameters. It is possible to reduce the size of a user private key by a third by discarding the final component. The user can recover the final component using the private key relation from clause 5.6, but they would not be able to check that the private key is valid by directly verifying that the relation holds.

Similarly, a user private key for two-level LATTE contains a quadruple of small elements from R. Discarding the final component reduces the size of a user private key size by a quarter, but again the user would not be able to check that the private key is valid by directly verifying that the key relation holds.

6.3.3.4 Ciphertext sizes

An uncompressed ciphertext for single-level LATTE contains a triple of elements from R_q and a 256-bit encrypted message so will be almost three times the size of a DLP ciphertext for similar parameters. Compressing the ciphertext by only sending the most significant bits of the final component could potentially reduce its size by up to a third, although the parameters would need to be adjusted to keep an acceptable decryption failure probability (see clause B.1). Compressing the ciphertext for two-level LATTE could potentially reduce its size by up to a quarter.

6.4 Security estimates

Tables 3 and 4 give summaries of the estimated classical and quantum costs of the different lattice attacks and heuristic upper bounds on the probability of decryption failure for the single-level parameter and two-level parameters listed in clause 6.3.

Table 3: Classical and quantum security estimates for single-level IBE schemes

Scheme	Master Key Recovery		User Key Recovery		Message Recovery		Decryption Failure
	(C)	*(Q)*	*(C)*	*(Q)*	*(C)*	*(Q)*	
LATTE-1	300 bits	274 bits	246 bits	224 bits	139 bits	128 bits	2^{-124}
LATTE-2	589 bits	536 bits	535 bits	487 bits	298 bits	272 bits	2^{-250}

Table 4: Classical and quantum security estimates for two-level HIBE schemes

Scheme	Master Key Recovery		Delegated Key Recovery		User Key Recovery		Message Recovery		Decryption Failure
	(C)	*(Q)*	*(C)*	*(Q)*	*(C)*	*(Q)*	*(C)*	*(Q)*	
LATTE-3	300 bits	274 bits	130 bits	120 bits	112 bits	103 bits	84 bits	77 bits	2^{-90}
LATTE-4	589 bits	536 bits	281 bits	257 bits	245 bits	224 bis	180 bits	164 bits	2^{-177}

The decryption failure probabilities given for LATTE-1 and LATTE-2 in Table 3 do not match the target security levels of 128 and 256-bits. However, these are loose upper bounds and the true probability of failure is likely to be much smaller. In all cases, the security of the parameter set is determined by the cost of the message recovery attack.

7 Performance estimates

7.1 Performance on a 64-bit desktop processor

Table 5 gives performance estimates (in milliseconds) for master key generation, user key extraction and delegation for DLP and LATTE on a 64-bit AMD A10-6700 quad-core desktop processor operating at 3.7 GHz and with Turbo Core disabled. The implementations were compiled using gcc version 7.4.0 in an Ubuntu® 18.04 virtual machine managed by Oracle® VirtualBox™ 5.2.26 with Microsoft® Windows® 8.1 as the host operating system. The compiler optimization flag was set to -O2 and the implementation was configured to allow the use of AVX instructions.

Table 5: Performance of key generation and extraction for DLP on a 64-bit desktop processor

Scheme	n	$\log_2 q$	Performance: Key Generation	Extraction	Notes
DLP-0	512	22	77 ms	1,3 ms	[i.33], see note
DLP-3	1 024	22	230 ms	2,5 ms	[i.33], see note
NOTE: The SAFEcrypto implementation of DLP [i.33] includes the efficient key generation method from [i.34] and the fast Fourier sampling technique from [i.35].					

Table 6 gives performance estimates (in milliseconds) for encryption and decryption for DLP and LATTE on the same platform. The compiler optimization flag was set to -O2 and for DLP the implementation was configured to allow the use of AVX instructions. The implementation of LATTE does not use AVX instructions.

Table 6: Performance of encryption and decryption on a 64-bit desktop processor

Scheme	n	$\log_2 q$	Performance: Level 1 Encryption	Level 1 Decryption	Level 2 Encryption	Level 2 Decryption	Notes
DLP-0	512	22	0,13 ms	0,053 ms	---	---	[i.33], note 1
DLP-3	1 024	22	0,43 ms	0,27 ms	---	---	[i.33], note 1
LATTE-1	1 024	24	0,39 ms	0,38 ms	---	---	note 2
LATTE-2	2 048	25	0,85 ms	0,84 ms	---	---	note 2
LATTE-3	1 024	36	0,60 ms	0,64 ms	0,78 ms	0,77 ms	note 2
LATTE-4	2 048	38	1,6 ms	1,7 ms	2,0 ms	2,1 ms	note 2
NOTE 1: The SAFEcrypto implementation of DLP [i.33] does not include ciphertext validation. NOTE 2: The implementation of LATTE uses the NTT as described in clause B.2.							

For comparison, the OpenQuantumSafe implementation of New Hope [i.36] with ring dimension $n = 1\,024$ and modulus $q = 2^{14} - 2^{12} + 1$ takes 0,21 ms for key generation, 0,28 ms for encapsulation and 0,28 ms for decapsulation on the same platform.

7.2 Performance on a 32-bit embedded processor

Table 7 gives performance estimates (in milliseconds) for encryption and decryption for master key generation, user key extraction and delegation for DLP and LATTE on a single board computer containing a 32-bit ARM1176™ core operating at 700 MHz. The implementations were compiled using gcc version 6.3.0 with the Debian® 9.1 operating system. The compiler optimization flag was set to -O2. AVX instructions are not supported by the ARM1176™ core.

Table 7: Performance of key generation and extraction for DLP on a 32-bit embedded processor

Scheme	n	$\log_2 q$	Performance: Key Generation	Extraction	Notes
DLP-0	512	22	740 ms	20 ms	[i.33], see note
DLP-3	1 024	22	2 400 ms	42 ms	[i.33], see note
NOTE: The SAFEcrypto implementation of DLP [i.33] includes the efficient key generation method from [i.34] and the fast Fourier sampling technique from [i.35].					

Table 8 gives performance estimates (in milliseconds) for encryption and decryption for DLP and LATTE on the same platform. The compiler optimization flag was set to -O2. AVX instructions are not supported by the ARM1176™ core.

Table 8: Performance of encryption and decryption on a 32-bit embedded processor

Scheme	n	$\log_2 q$	Performance: Level 1 Encryption	Level 1 Decryption	Level 2 Encryption	Level 2 Decryption	Notes
DLP-0	512	22	5,2 ms	2,5 ms	---	---	[i.33], note 1
DLP-3	1 024	22	15 ms	9,1 ms	---	---	[i.33], note 1
LATTE-1	1 024	24	24 ms	26 ms	---	---	note 2
LATTE-2	2 048	25	53 ms	57 ms	---	---	note 2
LATTE-3	1 024	36	26 ms	28 ms	34 ms	35 ms	note 2
LATTE-4	2 048	38	110 ms	130 ms	140 ms	160 ms	note 2
NOTE 1: The SAFEcrypto implementation of DLP [i.33] does not include ciphertext validation. NOTE 2: The implementation of LATTE uses the NTT as described in clause B.2.							

For comparison, the OpenQuantumSafe implementation of New Hope [i.36] with ring dimension $n = 1\,024$ and modulus $q = 2^{14} - 2^{12} + 1$ takes 15 ms for key generation, 22 ms for encapsulation and 19 ms for decapsulation on the same platform.

7.3 Discussion

7.3.1 Key generation

Master key generation for LATTE is almost identical to master key generation for DLP so its performance will be similar for similar parameters. However, performance is not critical as the master key pair needs to be generated once during the initial KMS setup and is unlikely to updated frequently, if at all, after that.

In the original implementation paper [i.20], McCarthy et al give a performance estimate of 2,7 seconds for DLP-1 and 17 seconds for DLP-3 on an Intel® Core™ i7 6700 running at 4 GHz. The improved figures in Table 5 are based on the updated SAFEcrypto implementation [i.33] which includes ideas from [i.34] to speed up the norm calculation in step 2, the resultant computations in steps 3 and 4, and the reduction in step 7.

Doubling the dimension of the ring increases the cost of key generation by between a factor of 4 and 8. Increasing the modulus also increases the cost of key generation, although the effect will be less significant. This means that the largest LATTE parameters will likely have master key generation times on the order of seconds on a 64-bit desktop or server processor.

7.3.2 Extraction

User key extraction for LATTE is based on key extraction for DLP and the most expensive step for both is the Klein sampler so the performance will be similar for similar parameters. Extraction needs to be efficient enough for the KMS or sub-KMS to extract keys for all the users it manages in a reasonable amount of time.

In the original implementation paper [i.20], McCarthy et al give a performance estimate of 1,7 milliseconds for DLP-1 and 7,4 milliseconds for DLP-3 on an Intel® Core™ i7 6700 running at 4 GHz. The figures in Table 5 are based on the updated SAFEcrypto implementation [i.33] which replaces the compact Klein sampler from [i.32] (see clause A.4) with the fast Fourier sampler from [i.35]. Although the fast Fourier sampler does provide a slight improvement over the compact sampler, the overall performance of key extraction depends more on the efficiency of the underlying one-dimensional discrete Gaussian sampler.

The compact Klein sampler is a quadratic algorithm so doubling the dimension of the ring will increase the cost of key extraction by around a factor of 4. Further, level 2 key extraction requires a higher level of floating-point precision than level 1 key extraction so will be 2 to 4 times more expensive. This means that largest LATTE parameters will likely have key extraction times on the order of tens or hundreds of milliseconds on a 64-bit desktop or server processor.

7.3.3 Delegation

Delegation for LATTE is a higher-rank analogue of DLP master key generation so will be substantially slower than key generation for similar parameters. Its performance is not critical as the delegated private keys need to be generated once for each sub-KMS during initial setup and are unlikely to updated frequently, if at all, after that.

The Klein sampler used in step 2.2 of delegation is exactly the same as the Klein sampler used in key extraction. The norm calculation in step 2.4 is straightforward and the techniques from [i.34] can be applied to the resultant computation in step 3.2 and the reduction in step 8. On the other hand, the rank has increased, the polynomials have larger coefficients and the floating-point precision is higher so delegation will probably be at least 4 times more expensive than key generation. This means that largest LATTE parameters will likely have delegation times on the order of minutes on a 64-bit desktop or server processor.

7.3.4 Encryption and decryption

Encryption and decryption for LATTE follow the general approach from the CCA-secure version of New Hope [i.31], but LATTE ciphertexts include more terms so it will be slightly slower. The performance of encryption and decryption is critically important, particularly on constrained devices.

The figures for LATTE in Tables 6 and 8 are taken from an implementation that uses the NTT approach described in clause B.2. These show that the performance of encryption and decryption for LATTE is within a factor of 2 of the performance of New Hope with similar parameters. Further, encryption and decryption for level 2 users does not seem to be significantly more expensive than encryption and decryption for level 1 users. The largest LATTE parameters have encryption and decryption times on the order milliseconds on a 64-bit desktop or server processor, and on the order of hundreds of milliseconds on a 32-bit embedded processor.

8 Conclusion

Identity-based and hierarchical identity-based encryption potentially offer useful functionality for enterprise and IoT applications, but most existing schemes rely on pairing-based cryptography which is vulnerable to quantum computers. Lattice-based cryptography can be used to construct quantum-safe alternatives that have been shown to be practical when instantiated with structured lattices.

The present document gives a high-level description of LATTE, one proposed approach to hierarchical identity-based encryption using structured lattices. It is built from well-known lattice techniques such as NTRU-style trapdoor lattices, Ring-LWE encryption, Klein sampling and bonsai trees. Consequently, it shares a number of advantages and disadvantages that are common to other lattice-based schemes.

For example:

- Encryption and decryption are straightforward to implement and are fast, even on constrained devices;
- Key generation, extraction and delegation can be optimized using the structure in cyclotomic rings; and
- The security proof provides a level of reassurance and the practical security is increasingly well understood.

On the other hand:

- Extraction and delegation require higher-precision arithmetic and can be slow, even on desktops or servers;
- Ciphertext sizes can be large and do not scale well as the number of levels in the hierarchy increases; and
- Side-channel and other implementation vulnerabilities are not yet well understood.

LATTE is still a relatively new scheme and will benefit from further research to improve performance, reduce bandwidth requirements and develop effective side-channel protections. Similarly, to mitigate some of the issues around the use of identity-based and hierarchical identity-based cryptography it is also worth investigating whether LATTE can support distributed key extraction, more efficient revocation mechanisms or mediated decryption.

Annex A: Mathematical background

A.1 Lattices

A.1.1 Bases and determinant

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a set of vectors in $\mathbb{R}^m$. The corresponding lattice is the integer linear span of $\mathcal{B}$; that is,

$$\mathcal{L}(\mathcal{B}) = \{x_1 b_1 + \cdots + x_n b_n : (x_1, \ldots, x_n) \in \mathbb{Z}^n\}.$$

If the vectors in $\mathcal{B}$ are linearly independent, then $\mathcal{B}$ is called a basis for the lattice and n is the dimension. The basis can be represented as a matrix $\mathcal{B} \in \mathbb{R}^{n \times m}$ with the vectors b_i as the rows.

A lattice $\mathcal{L}$ can have many different bases, but the volume of the fundamental parallelepiped determined by the basis vectors is independent of the choice of basis. This is called the determinant $\det(\mathcal{L})$ of the lattice and can be computed as

$$\det(\mathcal{L}) = \sqrt{\det(\mathcal{B}\mathcal{B}^T)}$$

given any basis $\mathcal{B}$ for $\mathcal{L}$.

If $\mathcal{B} = \{b_1, \ldots, b_n\}$ has been sorted so that $\|b_1\| \leq \|b_2\| \leq \cdots \leq \|b_n\|$, then a measure of the quality of a basis for the lattice is given by the root Hermite factor δ where

$$\|b_1\| = \delta^n \cdot \det(\mathcal{L})^{1/n}.$$

A.1.2 Gram-Schmidt

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a basis for a lattice $\mathcal{L}$. The corresponding Gram-Schmidt vectors $\mathcal{B}^* = \{b_1^*, \ldots, b_n^*\}$ are computed as follows:

1) For each $i \in \{1, \ldots, n\}$:
   1.1) Set $b_i^* = b_i$.
   1.2) For each $j \in \{1, \ldots, i-1\}$:
      1.2.1) Set $c_{i,j} = \langle b_i, b_j^* \rangle / \|b_j^*\|^2$.
      1.2.2) Set $b_i^* = b_i^* - c_{i,j} b_j^*$.
2) Return $\mathcal{B}^* = \{b_1^*, \ldots, b_n^*\}$.

The Gram-Schmidt norm of the basis $\mathcal{B}$ is the length of the largest Gram-Schmidt vector; that is,

$$\|\mathcal{B}\|_{\mathrm{GS}} = \max(\|b_1^*\|, \ldots, \|b_n^*\|).$$

A.1.3 Nearest plane algorithm

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a reduced basis for a lattice $\mathcal{L}$ with corresponding Gram-Schmidt vectors $\mathcal{B}^* = \{b_1^*, \ldots, b_n^*\}$ and let $t \in \mathbb{Z}^n$ be an integer-valued vector. The nearest plane algorithm finds a vector in the lattice $\mathcal{L}$ close to t (see Figure 2.5 in [i.24]):

1) Set $t_n = t$.
2) For each $i \in \{n, \ldots, 1\}$:
   2.1) Set $c_i = \langle t_i, b_i^* \rangle / \|b_i^*\|^2$.

2.2) Set $t_{i-1} = t_i - \lfloor c_i \rceil b_i$.

3) Return $v = t - t_0$.

A.2 Lattice-basis reduction

A.2.1 Gaussian heuristic

Let $\mathcal{L}$ be an n-dimensional lattice. The Gaussian heuristic

$$GH(\mathcal{L}) = \frac{(\Gamma(n/2+1) \cdot \det(\mathcal{L}))^{1/n}}{\sqrt{\pi}}$$

gives an estimate for the shortest non-zero vector in $\mathcal{L}$. Denote by $GH(n)$ the Gaussian heuristic for an n-dimensional lattice with determinant 1; that is,

$$GH(n) = \Gamma(n/2+1)^{1/n}/\sqrt{\pi}.$$

A.2.2 Estimating quality

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a basis for a lattice $\mathcal{L}$ after lattice-basis reduction. The Geometric Series Assumption states that the lengths of the Gram-Schmidt vectors can be approximated by

$$\|b_i^*\| \approx \alpha^{i-1}\|b_1^*\|$$

for some constant α which depends on the lattice-basis reduction algorithm used. Further, the product of the lengths of the Gram-Schmidt vectors is equal to the determinant of $\mathcal{L}$ which means that $\alpha \approx \delta^{-2}$.

For the BKZ lattice-basis reduction algorithm with block size β the root Hermite factor for the basis can be approximated by (see, for example,[i.37])

$$\delta \approx GH(\beta)^{1/(\beta-1)}.$$

Consequently, the lengths of the Gram-Schmidt vectors can be approximated by

$$\|b_i^*\| \approx GH(\beta)^{(n-2(i-1))/(\beta-1)} \cdot \det(\mathcal{L})^{1/n}.$$

A.2.3 Estimating lattice-basis cost

The classical cost of lattice-basis reduction with block size β is estimated as the cost of lattice sieving in dimension β. The asymptotic formula for the most efficient classical variant of sieving [i.38] is

$$2^{0.292\beta + o(\beta)}.$$

Following [i.39], the constant term in the asymptotic formula is estimated using the practical experiments in [i.38] to give a classical cost estimate of

$$2^{0.292\beta + 16.4}.$$

Similarly, the asymptotic formula for the most efficient quantum variant of sieving [i.40] is

$$2^{0.265\beta + o(\beta)}$$

and using the same constant term as above gives a quantum cost estimate of

$$2^{0.265\beta + 16.4}.$$

A.3 Sampling

A.3.1 Discrete Gaussians

The n-dimensional continuous Gaussian function with standard deviation $\sigma > 0$ centred at $c \in \mathbb{R}^n$ is defined to be the distribution over $\mathbb{R}^n$ with probability density function proportional to

$$\rho_{\sigma,c}(x) = e^{-\frac{\|x-c\|^2}{2\sigma^2}}.$$

The corresponding discrete Gaussian distribution over the lattice $\mathcal{L}$ in $\mathbb{R}^n$ is defined to be the distribution with probability density function

$$D_{\sigma,c,\mathcal{L}}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\mathcal{L})}.$$

In general, care needs to be taken when sampling polynomials from a discrete Gaussian [i.41]. However, for a discrete Gaussian over the power-of-two cyclotomic ring R, it is enough to sample coefficients independently from a one-dimensional Gaussian $D(\sigma, c_i)$ over $\mathbb{Z}$ with the same standard deviation (see, for example, Section 1 of [i.42]).

A.3.2 Klein sampler

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a reduced basis for a lattice $\mathcal{L}$ with corresponding Gram-Schmidt vectors $\mathcal{B}^* = \{b_1^*, \ldots, b_n^*\}$, let σ be a target standard deviation, and let $t \in \mathbb{Z}^n$ be an integer-valued vector. The Klein sampler produces vectors sampled from a distribution that is close to the discrete Gaussian $D_{\mathcal{L}}(t, \sigma)$ over $\mathcal{L}$ with standard deviation and centre t. This can be viewed as a randomized version of the nearest plane algorithm:

1) Set $t_n = t$.
2) For each $i \in \{n, \ldots, 1\}$:
   2.1) Set $c_i = \langle t_i, b_i^* \rangle / \|b_i^*\|^2$.
   2.2) Set $\sigma_i' = \sigma / \|b_i^*\|^2$.
   2.3) Sample $z_i \in \mathbb{Z}$ from a discrete Gaussian distribution $D(c_i, \sigma_i')$ with mean c_i and standard deviation σ_i'.
   2.4) Set $t_{i-1} = t_i - z_i b_i$.
3) Return $v = t - t_0$.

A.4 NTRU-style lattices

A.4.1 Isometric lattices

The NTRU-style lattices that are used in LATTE are defined using a basis of the form

$$\mathcal{B} = \begin{bmatrix} \mathcal{M}(b_{1,1}) & \cdots & \mathcal{M}(b_{1,k}) \\ \vdots & \ddots & \vdots \\ \mathcal{M}(b_{k,1}) & \cdots & \mathcal{M}(b_{k,k}) \end{bmatrix}$$

where each block $\mathcal{M}(b_{i,j})$ is the matrix corresponding to a polynomial $b_{i,j} \in R$. This means that multiplication by x is a lattice isometry: given a vector $v = (v_1, \ldots, v_k)$ in the lattice for some $v_1, \ldots, v_k \in R$, the vector

$$x \cdot v = (x \cdot v_1, \ldots, x \cdot v_k)$$

also lies in the lattice and has the same length as v. Further, the basis $\mathcal{B}$ can be written as $\mathcal{B} = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_k$ where

$$\mathcal{B}_i = \{(b_{i,1}, \ldots, b_{i,k}), x \cdot (b_{i,1}, \ldots, b_{i,k}), \ldots, x^{n-1} \cdot (b_{i,1}, \ldots, b_{i,k})\}.$$

Lyubashevsky and Prest [i.32] have shown that the block isometric structure of NTRU-style lattices can be used to give significantly more efficient versions of Gram-Schmidt orthogonalisation and Klein sampling.

A.4.2 Isometric Gram-Schmidt

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be an isometric basis for an n-dimensional lattice $\mathcal{L}$; that is, the basis vectors are $b_i = x^{i-1} \cdot b_1$ for each $i \in \{1, \ldots, n\}$. The isometric Gram-Schmidt algorithm computes the Gram-Schmidt vectors in time quadratic in n (see Algorithm 3 from [i.32]):

1) Set $b_1^* = b_1$.
2) Set $v_1 = b_1$.
3) Set $C_1 = \langle v_1, x \cdot b_1^* \rangle$.
4) Set $D_1 = \|v_1\|^2$.
5) For each $i \in \{2, \ldots, n\}$:
   5.1) Set $b_i^* = x \cdot b_{i-1}^* - (C_{i-1}/D_{i-1})\, v_{i-1}$.
   5.2) Set $v_i = v_{i-1} - (C_{i-1}/D_{i-1})\, x \cdot b_{i-1}^*$.
   5.3) Set $C_i = \langle v_1, x \cdot b_i^* \rangle$.
   5.4) Set $D_i = D_{i-1} - C_{i-1}^2/D_{i-1}$.
6) Return the Gram-Schmidt vectors $B^* = \{b_1^*, \ldots, b_n^*\}$.

A.4.3 Isometric Klein sampler

The isometric Gram-Schmidt algorithm described above computes the ith vector from the $(i-1)$st using the values C_{i-1} and D_{i-1}. It is possible to reverse the equations in steps 5.1 and 5.2 to compute the $(i-1)$st Gram-Schmidt vector from the ith vector using the values C_{i-1}, D_{i-1} and D_i. This means that the Klein sampler can be modified to compute the Gram-Schmidt vectors during the sampling process and so avoid the need to store all of them.

Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be an isometric basis for an n-dimensional lattice $\mathcal{L}$, let $C_1, \ldots, C_n$ and $D_1, \ldots, D_n$ be the scalars computed by the isometric Gram-Schmidt algorithm, and let b_n^* and v_n be the final vectors. Given a target standard deviation σ and vector $t \in \mathbb{Z}^n$, the isometric Klein sampler produces a vector sampled from a distribution that is close to the discrete Gaussian $D_{\mathcal{L}}(t, \sigma)$ over $\mathcal{L}$ (see Algorithm 9 from [i.32]):

1) Set $t_n = t$.
2) For each $i \in \{n, \ldots, 1\}$:
   2.1) Set $c_i = \langle t_i, b_i^* \rangle / \|b_i^*\|^2$.
   2.2) Set $\sigma_i' = \sigma / \|b_i^*\|^2$.
   2.3) Sample $z_i \in \mathbb{Z}$ from a discrete Gaussian distribution $D(c_i, \sigma_i')$ with mean c_i and standard deviation σ_i'.
   2.4) Set $t_{i-1} = t_i - z_i b_i$.
   2.5) Set $v_{i-1} = (D_{i-1}/D_i)(x^{-1} \cdot b_i^*) + (C_{i-1}/D_i)(x^{-1} \cdot v_i)$.
   2.6) Set $b_{i-1}^* = (C_{i-1}/D_i) b_i^* + (D_{i-1}/D_i) v_i$.
3) Return $v = t - t_0$.

A.4.4 Block isometric Gram-Schmidt

Let $\mathcal{B} = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_k$ be a block isometric basis for a kn-dimensional lattice $\mathcal{L}$ where the basis vectors in $\mathcal{B}_i$ are

$$\mathcal{B}_i = \left\{b_{i,1}, b_{i,2} = x \cdot b_{i,1}, \ldots, b_{i,n} = x^{n-1} \cdot b_{i,1}\right\}$$

for each $i \in \{1, \ldots, k\}$. The block isometric Gram-Schmidt algorithm computes the Gram-Schmidt vectors by using the isometric Gram-Schmidt algorithm from clause A.4.2 on each block in turn (see Algorithm 5 from [i.32]):

1) For each $i \in \{1, \ldots, k\}$:

   1.1) Set $b^*_{(i-1)n+1} = b_{(i-1)n+1}$.

   1.2) For each $j \in \{1, \ldots, (i-1)n\}$:

      1.2.1) Set $b^*_{(i-1)n+1} = b^*_{(i-1)n+1} - (\langle b_{(i-1)n+1}, b^*_j \rangle / \left\|b^*_j\right\|^2) b^*_j$.

   1.3) Compute $\mathcal{B}^*_i$ for the block with isometric basis $\left\{b^*_{(i-1)n+1}, x \cdot b^*_{(i-1)n+1}, \ldots, x^{n-1} \cdot b^*_{(i-1)n+1}\right\}$.

2) Return the Gram-Schmidt vectors $\mathcal{B}^* = \mathcal{B}^*_1 \cup \cdots \cup \mathcal{B}^*_k$.

A.4.5 Block isometric Klein sampler

Let $\mathcal{B} = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_k$ be a block isometric basis for a kn-dimensional lattice $\mathcal{L}$, let $C_1, \ldots, C_{kn}$ and $D_1, \ldots, D_{kn}$ be the scalars computed by the block isometric Gram-Schmidt algorithm, and let $b^*_n, \ldots, b^*_{kn}$ and $v_n, \ldots, v_{kn}$ be the final vectors computed in each block. Given a standard deviation σ and vector $t \in \mathbb{Z}^n$, the block isometric Klein sampler produces a vector sampled from a distribution that is close to the discrete Gaussian $D_{\mathcal{L}}(t, \sigma)$ over $\mathcal{L}$ by using the isometric Klein sampler from clause A.4.3 on each block:

1) Set $t_{kn} = t$.

2) For each $i \in \{k, \ldots, 1\}$:

   2.1) Sample $t_{(i-1)n}$ using the isometric Klein sampler on the block $\mathcal{B}_i$ with centre t_{in}.

3) Return $v = t - t_0$.

Annex B: Implementation considerations

B.1 Ciphertext compression

In decryption, the user computes

$$\begin{aligned} V &\equiv C_{\ell+1} - C_0 \cdot t_0 - \cdots - C_\ell \cdot t_\ell && (\mathrm{mod}\ q) \\ &\equiv m + (e_{\ell+1} - e_0 \cdot t_0 - \cdots - e_\ell \cdot t_\ell + e \cdot t_{\ell+1}) && (\mathrm{mod}\ q) \end{aligned}$$

and decodes m. This succeeds provided that the error term $d = e_{\ell+1} - e_0 \cdot t_0 - \cdots - e_\ell \cdot t_\ell + e \cdot t_{\ell+1}$ is sufficiently small.

The inherent noise in decryption means that it is possible to compress the ciphertext by only sending the most significant bits of each coefficient. For example, the initiator can compress the final ciphertext component $C_{\ell+1}$ by computing

$$\check{C}_{\ell+1} \equiv \lfloor 2^b C_{\ell+1}/q \rceil \pmod{2^b}$$

for some choice of b. During decryption, the user will need to decompress $\check{C}_{\ell+1}$ by computing

$$\hat{C}_{\ell+1} \equiv \left\lfloor q \check{C}_{\ell+1}/2^b \right\rceil \pmod{q}$$

which effectively adds a bounded uniform error to $C_{\ell+1}$. This increases the probability of decryption failure, but the modulus can be adjusted to compensate.

More generally, [i.43] suggests that every component of the ciphertext can be compressed in a similar manner.

NOTE: Ciphertext compression relies on the fact that small amounts of noise in the ciphertext coefficients only generate small errors in the decryption process. This means that ciphertext compression is not compatible with sending the ciphertext in the NTT domain (see clause B.2) since the NTT does not preserve size.

B.2 Number-Theoretic Transform

The Number Theoretic Transform is a variant of the discrete Fourier Transform which allows efficient arithmetic in the ring R_q (see, for example, [i.44]). More precisely, if $q \equiv 1 \pmod{2n}$, then the NTT is an efficiently computable ring isomorphism

$$\mathrm{NTT}\colon R_q \longrightarrow \mathbb{Z}/q\mathbb{Z} \times \cdots \times \mathbb{Z}/q\mathbb{Z}.$$

As it is a ring isomorphism, for any polynomials $a, b \in R_q$

$$\mathrm{NTT}(a \cdot b) \equiv \mathrm{NTT}(a) * \mathrm{NTT}(b),$$

where $*$ denotes co-ordinatewise multiplication. Polynomial multiplication in the ring can therefore be computed via

$$a \cdot b \equiv \mathrm{NTT}^{-1}\big(\mathrm{NTT}(a) * \mathrm{NTT}(b)\big)$$

which requires two forward NTTs, a co-ordinatewise multiplication and an inverse NTT.

Encryption and decryption can be modified to minimize the number of times that the NTT or its inverse is applied:

- the master public key and user private keys can be stored in the NTT domain;
- the hash function can be used to generate polynomials directly in the NTT domain; and
- the ciphertext can be transmitted in the NTT domain. [i.44]

A simplified version of encryption for the single-level LATTE might proceed as follows:

1) The initiator computes the hash $\mathrm{NTT}(A_1) = H(ID_1)$ directly in the NTT domain.

2) The initiator samples polynomials $e, e_0, e_1, e_2 \in R$ and computes their transforms $\mathrm{NTT}(e)$, $\mathrm{NTT}(e_0)$, $\mathrm{NTT}(e_1)$ and $\mathrm{NTT}(e_2)$.

3) The initiator encodes the message as a polynomial $m \in R$ with coefficients from $\{0, (q-1)/2\}$ and computes its transform $\mathrm{NTT}(m)$.

4) The initiator constructs the ciphertext directly in the NTT domain by computing

$$\mathrm{NTT}(C_0) \equiv \mathrm{NTT}(A) * \mathrm{NTT}(e) + \mathrm{NTT}(e_0)$$
$$\mathrm{NTT}(C_1) \equiv \mathrm{NTT}(A_1) * \mathrm{NTT}(e) + \mathrm{NTT}(e_1)$$
$$\mathrm{NTT}(C_2) \equiv \mathrm{NTT}(B) * \mathrm{NTT}(e) + \mathrm{NTT}(e_2) + \mathrm{NTT}(m)$$

where $\mathrm{NTT}(A)$ and $\mathrm{NTT}(B)$ are the two components of the master public key which have been stored in the NTT domain.

5) The initiator sends the ciphertext $(\mathrm{NTT}(C_0), \mathrm{NTT}(C_1), \mathrm{NTT}(C_2))$ to the user in the NTT domain.

This approach saves three forward NTTs and three inverse NTTs.

The corresponding decryption process would then be as follows:

1) The user computes the intermediate value V in the NTT domain

$$\mathrm{NTT}(V) \equiv \mathrm{NTT}(C_2) - \mathrm{NTT}(C_0) * \mathrm{NTT}(t_0) - \mathrm{NTT}(C_1) * \mathrm{NTT}(C_1)$$

where $\mathrm{NTT}(t_0)$ and $\mathrm{NTT}(t_1)$ are the first two components of the user private key which have been stored in the NTT domain.

2) The user computes the inverse transform $V = \mathrm{NTT}^{-1}(\mathrm{NTT}(V))$ and decodes V to recover m.

This approach saves five forward NTTs, but with ciphertext validation it would save a total of eight forward NTTs.

NOTE 1: Size is not preserved by the NTT so sending the ciphertext in the NTT domain is incompatible with the compression techniques discussed in clause B.1.

NOTE 2: There are several different variants of the NTT algorithm, so the choice of algorithm would need to be fixed to ensure interoperability when sending the ciphertext, hashing the identifiers, or storing the master public keys in the NTT domain.

Annex C: Security considerations

C.1 Provable security

C.1.1 Security definitions

LATTE is a modification of the generic hierarchical identity-based key encapsulation mechanism (KEM) from [i.15].

KEMs differ slightly from encryption schemes when used for transporting a symmetric key. In an encryption scheme, the symmetric key is chosen by the initiator and given as an input to the encryption operation. In a KEM, the symmetric key is derived during encapsulation and returned as an output. Nevertheless, the two approaches have very similar security properties.

The usual passive security notion for public-key encryption schemes is indistinguishability under chosen-plaintext attack (IND-CPA):

1) The challenger generates a key pair (sk, pk).
2) The adversary is given the public key pk and chooses a pair of symmetric keys k_0 and k_1.
3) The challenger selects one of k_0 or k_1 and encrypts it using the public key pk to produce a ciphertext Z.
4) The adversary is given the ciphertext Z and asked to guess which of k_0 or k_1 was encrypted.

An adversary $\mathcal{A}$'s advantage is

$$\mathrm{Adv}_{\mathrm{PKE}}(\mathcal{A}) = |p_1(\mathcal{A}) - p_0(\mathcal{A})|$$

where $p_1(\mathcal{A})$ is the probability that the adversary guesses correctly and $p_0(\mathcal{A})$ is the probability that it guesses incorrectly.

The corresponding notion for HIBEs (see Section 2.2 of [i.15]) is indistinguishability under chosen-plaintext adaptive-identity attack (ID-IND-CPA):

1) The challenger generates a master key pair (sk, pk).
2) The adversary is given the master pubic key pk and allowed to query oracles for delegation and extraction.
3) The adversary chooses a target sequence of identifiers $(ID_1^*, \ldots, ID_\ell^*)$, where the inputs to the oracle queries did not include $(ID_1^*, \ldots, ID_i^*)$ for $i \leq \ell$, and a pair of symmetric keys k_0 and k_1.
4) The challenger selects one of k_0 or k_1 and encrypts it using the target sequence $(ID_1^*, \ldots, ID_\ell^*)$ to produce a ciphertext Z.
5) The adversary is given Z and is allowed further queries to the delegation and extraction oracles with the restriction that the inputs cannot include $(ID_1^*, \ldots, ID_i^*)$ for any $i \leq \ell$.
6) The adversary is asked to guess which of k_0 or k_1 was encrypted.

The adversary $\mathcal{A}$'s advantage $\mathrm{Adv}_{\mathrm{HIBE}}(\mathcal{A})$ is defined in the same way as above.

C.1.2 Bonsai scheme

The master key generation and user key extraction steps for LATTE differ slightly from the hierarchical identity-based KEM from [i.15].

In the bonsai scheme, the master public key is a single element $A \in R_q$ and user key extraction aims to find a short solution $t_0, \ldots, t_{\ell+1} \in R$ to the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \equiv B \pmod{q}$$

where $A_0 = A$ is the master public key, $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$, and $B = G(ID_1 \,||\, \ldots \,||\, ID_\ell)$ for a second hash function $G : \{0,1\}^* \to R_q$.

This construction allows the authors of [i.15] to give a security proof for the bonsai scheme in the programmable random oracle model. Specifically, they show that given an adversary $\mathcal{A}$ for HIBE scheme there is a related adversary $S^{\mathcal{A}}$ for the underlying public-key encryption scheme and bound the advantage by

$$\mathrm{Adv}_{\mathrm{HIBE}}(\mathcal{A}) \leq d Q_H^{d-1} Q_G \mathrm{Adv}_{\mathrm{PKE}}(S^{\mathcal{A}}) + \mathrm{negl}(n)$$

where Q_H is the number of queries $\mathcal{A}$ makes to the random oracle H and Q_G is the number of queries $\mathcal{A}$ makes to the random oracle G.

The proof relies on the construction of a simulator $S^{\mathcal{A}}$ that is able to answer delegation and key extraction queries without access to the private key sk.

- When the oracle H is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator uses the master key generation process to construct an element $A_\ell \in R$ so that the corresponding lattice has a known short basis, and programs the output of the oracle to be $H(ID_1 \,||\, \ldots \,||\, ID_\ell) = A_\ell$.

- When the delegation oracle is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator uses the known short basis for the lattice corresponding to $H(ID_1 \,||\, \ldots \,||\, ID_\ell) = A_\ell$ to produce a delegated basis for the extended lattice in the same way as normal delegation.

- When the oracle G is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator chooses small elements $t_0, \ldots, t_{\ell+1} \in R$ and programs the output of the oracle to be $G(ID_1 \,||\, \ldots \,||\, ID_\ell) = B$ where

$$B \equiv A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \pmod{q}.$$

- When the extraction oracle is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator simply returns the sequence of small elements $(t_0, \ldots, t_{\ell+1})$ used to construct B.

NOTE: The security proof assumes that the master public keys are indistinguishable from random.

C.1.3 LATTE

In LATTE, key extraction aims to find a short solution $t_0, \ldots, t_{\ell+1} \in R$ to the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \equiv B \pmod{q}$$

where $A_0 = A$ is the first component of the master public key, $A_i = H(ID_1 \,||\, \ldots \,||\, ID_i)$ for $i \in \{1, \ldots, \ell\}$, and B is the second component of the master public key.

The simulator above can be modified to produce a similar security proof for LATTE by adjusting the way it handles queries to the extraction oracle.

- When the oracle H is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator uses the master key generation process to construct an element $A_\ell \in R$ so that the corresponding lattice has a known short basis, and programs the output of the oracle to be $H(ID_1 \,||\, \ldots \,||\, ID_\ell) = A_\ell$.

- When the delegation oracle is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator uses the known short basis for the lattice corresponding to $H(ID_1 \,||\, \ldots \,||\, ID_\ell) = A_\ell$ to produce a delegated basis for the extended lattice in the same way as normal delegation.

- When the extraction oracle is queried with input $(ID_1, \ldots, ID_\ell)$, the simulator uses the known short basis for the lattice corresponding to $H(ID_1 \,||\, \ldots \,||\, ID_\ell) = A_\ell$ to find a short solution $t_0, \ldots, t_{\ell+1} \in R$ to the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \equiv B \pmod{q}$$

in the same way as normal extraction.

The removal of the random oracle G from LATTE means that the bound on the advantage for an adversary $\mathcal{A}$ becomes

$$\mathrm{Adv}_{\mathrm{HIBE}}(\mathcal{A}) \leq dQ_H^{d-1}\mathrm{Adv}_{\mathrm{PKE}}(S^{\mathcal{A}}) + \mathrm{negl}(n)$$

where Q_H is the number of queries $\mathcal{A}$ makes to the random oracle H. This is tight for the single-level IBE scheme.

C.1.4 Active security

Indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA) is a stronger security notion where the adversary also has access to a decryption oracle which can be queried with any input except for the two challenge symmetric keys. The Fujisaki-Okamoto transform [i.28] is a standard technique for converting an IND-CPA secure public-key encryption scheme into one that is IND-CCA secure in the random oracle model. A discussion of the transform in the context of Ring-LWE encryption is given in [i.45].

The Fujisaki-Okamoto transform can be applied directly to an ID-IND-CPA secure HIBE scheme to provide a scheme that is ID-IND-CCA secure. LATTE uses a slight variant of the transform from [i.30].

C.2 Practical security

C.2.1 Statistical security

The Klein sampler uses a short basis for the lattice $\mathcal{L}$ to produce a vector sampled from a distribution that is close to the discrete Gaussian $D_{\mathcal{L}}(t, \sigma)$ over $\mathcal{L}$. If the distribution of samples is not sufficiently close to the target discrete Gaussian distribution, then the sampler can potentially leak information about the short basis [i.23].

The accuracy of the Klein sampler depends on the size of the standard deviation σ relative to the Gram-Schmidt norm of the basis $\mathcal{B}$. Theorem 2 of [i.18] states that if $\sigma \geq \eta'_\varepsilon(\mathbb{Z})\|\mathcal{B}\|_{GS}$, where

$$\eta'_\varepsilon(\mathbb{Z}) \approx \frac{1}{\pi}\sqrt{\frac{\ln(2 + 2/\varepsilon)}{2}}$$

is the smoothing parameter, then the Kullback-Leibler divergence of the Klein sampler from the discrete Gaussian is bounded above by

$$2\left(1 - \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^m\right)^2 \approx 8m^2\varepsilon^2,$$

where m is the dimension of the lattice. Choosing $\varepsilon = 2^{-22.5}/(\ell + 1)n$ ensures that the divergence of the sampler from the discrete Gaussian will be at most 2^{-48}.

The private basis $\mathcal{B}_\ell$ for a KMS at level ℓ will have Gram-Schmidt norm

$$\|\mathcal{B}_\ell\|_{GS} \leq \sqrt{(\ell + 2)n}\sigma_\ell$$

so the standard deviation $\sigma_{\ell+1}$ for delegation or extraction at level $\ell + 1$ is chosen such that

$$\sigma_{\ell+1} \geq \eta'_\varepsilon(\mathbb{Z})\sqrt{(\ell + 2)n}\sigma_\ell.$$

NOTE: Key generation and delegation will naturally produce bases with $\|\mathcal{B}_\ell\|_{GS} \approx \sqrt{(\ell + 2)n}\sigma_\ell$. Explicit norm checks are performed to ensure that $\|\mathcal{B}_\ell\|_{GS} \leq \sqrt{(\ell + 2)n}\sigma_\ell$.

C.2.2 Decryption failure

The Fujisaki-Okamoto transform provides security against active attacks that use malformed ciphertexts. However, to protect against active attacks that use random decryption failures it is important that the probability of a decryption failure is close to the target security level [i.46]. The analysis of the failure probability follows the general approach in [i.47].

The error term d in decryption at level ℓ is

$$d = e_{\ell+1} - e_0 \cdot t_0 - \cdots - e_\ell \cdot t_\ell + e \cdot t_{\ell+1},$$

where the t_i have coefficients sampled from a discrete Gaussian with standard deviation σ_ℓ and the e_i have coefficients sampled from an approximate discrete Gaussian with standard deviation σ_e. This means that coefficients of d can be modelled as Gaussian with standard deviation

$$\tau = \sqrt{\sigma_e^2 + (\ell + 2)n\sigma_\ell^2\sigma_e^2}.$$

Decryption will fail if for some $i \in \{0, \ldots, 255\}$

$$|d_{iu}| + \cdots + \left|d_{i(u+1)-1}\right| > uq/4.$$

By the special case of the Chernoff-Cramer bound given in [i.47], the probability that

$$\langle d, v \rangle \geq k$$

for any $v \in \mathbb{R}^n$ is less than $e^{-k^2/2\tau^2\|v\|^2}$. Restricting to vectors v with entries $v_{iu}, \ldots, v_{i(u+1)-1} \in \{1, -1\}$ and 0 elsewhere implies the probability that

$$|d_{iu}| + \cdots + \left|d_{i(u+1)-1}\right| = \max_v \langle d, v \rangle > k$$

for a fixed $i \in \{0, \ldots, 255\}$ is less than $2^u e^{-k^2/2\tau^2 u}$.

Consequently, the probability that the bound is exceeded for any $i \in \{0, \ldots, 255\}$ is less than $2^{u+8} e^{-k^2/2\tau^2 u}$.

C.2.3 Master key recovery

The security of the master KMS depends on the difficulty of recovering the unusually short vector (g, f) from the master lattice $\mathcal{L}_0$ given the public basis

$$\mathcal{B}_0 = \begin{bmatrix} qI_n & 0_n \\ \mathcal{M}(A) & I_n \end{bmatrix}.$$

The analysis of key recovery for the FALCON signature scheme [i.48] can be applied directly.

Lattice reduction with block size β will find the target vector provided that its projection onto the vector space spanned by the final β Gram-Schmidt vectors is shorter than length of the $(2n - \beta + 1)$-st Gram-Schmidt vector. This corresponds to

$$\sigma_0\sqrt{\beta} \leq GH(\beta)^{(2\beta-2n)/(\beta-1)} \cdot \det(\mathcal{L}_0)^{1/2n}.$$

As $\det(\mathcal{L}_0) = q^n$ and $\sigma_0 \approx \sqrt{qe/4n}$ the inequality only depends on n.

Table C.1 lists the minimal block sizes required for a successful master KMS attack and the corresponding cost of the lattice reduction.

Table C.1: Estimated cost of master key recovery

n	β	Classical security	Quantum security
1 024	972	300	273
2 048	1 960	588	535
4 096	3 943	1 167	1 061

NOTE: Although the modulus for LATTE is much larger than in NTRU, so is the standard deviation of the master private keys. This means that the "overstretched NTRU" attacks, such as [i.49], are not relevant.

C.2.4 Delegated key recovery

Delegation to a sub-KMS at level ℓ involves finding several sufficiently short vectors $(s_{i,0}, \ldots, s_{i,\ell+1})$ in the lattice $\mathcal{L}_\ell$ with public basis

$$\mathcal{B}_\ell = \begin{bmatrix} qI_n & 0_n & 0_n & \cdots & 0_n \\ \mathcal{M}(A) & I_n & 0_n & & 0_n \\ \mathcal{M}(A_1) & 0_n & I_n & & 0_n \\ \vdots & & & \ddots & \\ \mathcal{M}(A_\ell) & 0_n & 0_n & & I_n \end{bmatrix}.$$

However, for an attacker to act as a sub-KMS at level ℓ it is enough to set $s_2 = \cdots = s_{\ell+1} = 0$ and find a single vector (s_0, s_1) of length $\sigma_\ell \cdot \sqrt{2n}$ in the master lattice $\mathcal{L}_0$.

Lattice reduction with block size β will find a vector of the required length provided that it is longer than the first Gram-Schmidt vector. This corresponds to

$$\sigma_\ell \cdot \sqrt{2n} \geq GH(\beta)^{2n/(\beta-1)} \cdot \det(\mathcal{L}_0)^{1/2n}.$$

Note that since $\det(\mathcal{L}_0) = q^n$ and $\sigma_\ell \approx \sqrt{(\ell+1)n} \cdot \sigma_{\ell-1}$ with $\sigma_0 \approx \sqrt{qe/4n}$, this only depends on n and ℓ.

Table C.2 lists the minimal block sizes required for a successful sub-KMS attack and the corresponding cost of the lattice reduction.

Table C.2: Estimated cost of delegated key recovery

n	ℓ	β	Classical security	Quantum security
1 024	1	1 004	309	282
	2	387	129	118
2 048	1	2 209	661	601
	2	905	280	256
	3	515	166	152
4 096	2	2 015	604	550
	3	1 187	363	330
	4	803	250	229

C.2.5 User key recovery

Extraction of a user private key at level ℓ involves solving a close vector problem in the lattice $\mathcal{L}_\ell$ in order to find a sufficiently short solution $(t_0, \ldots, t_{\ell+1})$ to the equation

$$A_0 \cdot t_0 + A_1 \cdot t_1 + \cdots + A_\ell \cdot t_\ell + t_{\ell+1} \equiv B \pmod{q}$$

over R_q.

However, for an attacker to be able to decrypt any message sent to the user it is enough to set $t_1 = \cdots = t_\ell = 0$ and solve a close vector problem in the master lattice $\mathcal{L}_0$ to find a sufficiently short solution $(t_0, t_{\ell+1})$ to the equation

$$A \cdot t_0 + t_{\ell+1} \equiv B \pmod{q}.$$

If $(t_0, t_{\ell+1})$ has length $\sigma_\ell \cdot \sqrt{2n}$, then the probability of a decryption failure from the user key attack will be less than the corresponding probability for an honest user at level ℓ. Indeed, when attacking a user at the maximum level d in the hierarchy, finding $(t_0, t_{\ell+1})$ with length $\sigma_d \cdot \sqrt{2(d+2)} \cdot \sqrt{2n}$ will still give a reasonably low probability of decryption failure.

Table C.3 lists the minimal block sizes required for a successful user attack and the corresponding cost of the lattice reduction.

Table C.3: Estimated cost of user key recovery

n	d	β	Classical security	Quantum security
1024	1	775	242	221
	2	326	111	102
2048	1	1 757	529	482
	2	782	244	223
	3	459	150	138
4096	2	1 771	533	485
	3	1075	330	301
	4	739	232	212

C.2.6 Message recovery

The main two attacks against the ciphertexts are the primal key recovery attack and dual distinguishing attack as outlined in [i.47].

In the primal attack, the ephemeral private keys are recovered via a close vector problem in the $(\ell + 2)n$-dimensional lattice

$$\begin{bmatrix} qI_n & 0_n & \cdots & 0_n & 0_n \\ 0_n & qI_n & & 0_n & 0_n \\ \vdots & & \ddots & & \\ 0_n & 0_n & & qI_n & 0_n \\ \mathcal{M}(A) & \mathcal{M}(A_1) & & \mathcal{M}(A_\ell) & I_n \end{bmatrix}$$

with target vector $v = (C_0, \ldots, C_\ell, 0)$ since the corresponding error vector will be $(-e_0, \ldots, -e_\ell, e)$. As it is enough to recover e, the dimension of the lattice can be reduced by considering only the last $m + n$ rows and columns. Further, the close vector problem can be embedded as a unique short vector problem in an $(m + n + 1)$-dimensional lattice with determinant q^m.

Lattice reduction with block size β will find the short vector provided that its projection onto the vector space spanned by the final β Gram-Schmidt vectors is less than the length of the $(m + n - \beta + 2)$-nd Gram-Schmidt vector. This corresponds to

$$\sigma_e\sqrt{\beta} \leq GH(\beta)^{(2\beta-m-n-1)/(\beta-1)} \cdot q^{m/(m+n+1)}.$$

Table C.4 lists the minimal block size β and number of samples m for a successful primal message recovery attack and the corresponding cost of the lattice reduction.

Table C.4: Estimated cost of primal key recovery attack

n	$\log_2 q$	σ_e	m	β	Classical security	Quantum security
1024	24	2.0	1 017	423	140	128
	36	2.0	998	232	84	77
2048	25	2.0	1 962	967	299	272
	38	2.0	2 036	561	180	165

NOTE: The standard deviation for the ephemeral private keys is fixed to be $\sigma_e = 2.0$. This means that hybrid lattice attacks that exploit small private keys, such as [i.50], are not relevant.

In the dual attack, short vectors in the $(\ell + 2)n$-dimensional scaled dual lattice

$$\begin{bmatrix} qI_n & 0_n & 0_n & \cdots & 0_n \\ \mathcal{M}(A)^{\mathrm{tr}} & I_n & 0_n & & 0_n \\ \mathcal{M}(A_1)^{\mathrm{tr}} & 0_n & I_n & & 0_n \\ \vdots & & & \ddots & \\ \mathcal{M}(A_\ell)^{\mathrm{tr}} & 0_n & 0_n & & I_n \end{bmatrix}$$

are used to distinguish the ciphertext elements from uniformly random polynomials in R_q. As it is enough to distinguish the first ciphertext element C_0 from random, the dimension of the lattice can be reduced by considering only the first $m + n$ rows and columns.

Lattice reduction with block size β will produce vectors of length

$$l = GH(\beta)^{(m+n)/(\beta-1)} \cdot q^{n/(m+n)}.$$

These will be short enough to distinguish with advantage ε provided that

$$-\ln(\varepsilon/4) \leq 2\pi^2 l^2 \sigma_e^2 / q^2.$$

Finally, the number of times that the lattice reduction needs to be repeated is

$$\max(1, 1/(2^{0.2075\beta}\varepsilon^2))$$

where $2^{0.2075\beta}$ is the number of short vectors returned by lattice sieving (see section 6.4 of [i.47] for details).

Table C.5 lists the minimal block size β and number of samples m for a successful dual distinguishing attack and the corresponding cost of the lattice reduction.

Table C.5: Estimated cost of dual distinguishing attack

n	$\log_2 q$	σ_e	m	β	Classical security	Quantum security
1024	24	2.0	1 036	422	139	128
	36	2.0	1 005	232	84	77
2048	25	2.0	1 973	964	298	272
	38	2.0	2 101	560	180	164

History

Document history		
V1.1.1	December 2019	Publication