Abstract

This note describes a key exchange method based on the ring-LWE (RLWE) assumption. It builds upon several results, including Regev’s landmark quantum reduction from certain worst case lattice problems (approx. GapSVP and SIVP) to random instances of the search variant of a particular learning problem (LWE). It also builds on the follow on work of Lyubashevsky, Peikert and Regev on the average case hardness of the RLWE search variant for polynomially bounded numbers of RLWE samples, along with novel applications of automorphism groups in number fields for a RLWE search to decision reduction (thereby demonstrating pseudorandomness of RLWE in these number fields). Subsequently, these results were adopted for the construction of Diffie-Hellman like key exchange methods by Peikert, and then by Lindner and Peikert followed by Ding and then by Ding, Xie and Lin who proposed efficient variants of such protocols. Subsequent work by Peikert proposed another efficient variant, phrased as a key encapsulation method, incorporating a low bandwidth "reconciliation" technique allowing two parties to exactly agree on a uniformly distributed secret value from noisy RLWE instances. This was followed by a concrete instantiation with parameter sets by Bos, Costello, Naehrig, Stebila, followed by another instantiation by Alkim, Ducas, Poppelmann and Schwabe with the same ring polynomial but a smaller modulus and a different reconciliation method. Unlike most other public key cryptography based key exchange methods, it is believed that RLWE based key exchange would remain secure in the event that an adversary is able to build a quantum computer.

This document is a product of the Crypto Forum Research Group (CFRG).
and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24th, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License. Also refer to the subsequent section entitled "Code Implementations and Licensing".

Table of Contents

1. Introduction........................................3
2. Notation, Definitions and Operators.................4
   2.1 Algebra and Number Theory
   2.2 Set Notation
   2.3 Random Variables and Distributions
   2.4 Arithmetic
   2.5 Miscellaneous
3. Error Distribution and Embeddings..................11
   3.1 Sampling
4. Functions...........................................13
4.1 Modular Rounding Function
4.2 Cross Rounding Function
4.3 Randomized Doubling Function
4.4 Reconciliation Function.
4.5 Extending to the rings OK_q and OK_2q
  4.5.1 Extended Modular Rounding Function
  4.5.2 Extended Cross Rounding Function
  4.5.3 Extended Reconciliation Function
  4.5.4 Extended Randomized Doubling Function

5. Flow Diagram.................................16


7. Protocol........................................17
  7.1 Server
    7.1.1 Ring Element ‘a’
    7.1.2 Server Secret Key and Error
    7.1.3 Server Public Key
    7.1.4 Server Key and Parameter Exchange
    7.1.5 Client Key and Cross Rounding Vector Receipt
    7.1.6 Server Reconciliation and Key Derivation
  7.2 Client
    7.2.1 Client Secret Key and Error
    7.2.2 Server Key and Parameter Receipt
    7.2.3 Client Public Key
    7.2.4 Client Doubling and Cross Rounding
    7.2.5 Client Key and Cross Rounding Vector Exchange
    7.2.6 Client Modular Rounding and Key Derivation

8. TLS Extensions.................................20
  8.1 Fundamental RLWE Suites
  8.2 Fundamental RLWE Key Exchange Algorithms
    8.2.1 RLWE_RSA
    8.2.2 RLWE_ECDSA
  8.3 Fundamental TLS Extensions for RLWE
    8.3.1 Fundamental RLWE Parameters Extension
    8.3.2 Client Hello Extension
    8.3.3 Server Hello Extension
    8.3.4 Server Certificate
    8.3.5 Server Key Exchange
    8.3.6 Certificate Request
    8.3.7 Client Certificate
    8.3.8 Client Key Exchange
    8.3.9 Certificate Verify
  8.4 Hybrid TLS Extensions for RLWE

9. Authenticity.................................28

10. Code Implementations and Licensing.............28

11. IANA Considerations........................29

12. Security Considerations....................29
  12.1 Security Parameters
  12.2 Ring Element ‘a’
1. Introduction

Recent years have seen the development of key exchange protocols from problems on lattices (discrete additive subgroups in vector spaces). This note describes one such method based on the ring-LWE problem. We start by describing some of the main results connecting lattice problems with lattice based cryptosystems. Regev [Reg2005] demonstrated a quantum reduction from certain worst case lattice problems (approx. GapSVP and SIVP) to random instances of the search variant of a particular learning problem (LWE). This means that an oracle that returns the LWE secret vector from randomly selected LWE instances implies an efficient quantum algorithm for approximating lattice problems. Peikert demonstrated a classical reduction from approx. GapSVP to decision LWE, partially de-quantizing Regev’s result [Pei2009a]. This reduction, however, does not extend to approx. SIVP and requires a modulus exponential in the lattice dimension. Lyubashevsky, Peikert and Regev [LPR2013a] used Regev’s quantum reduction along with aspects and tools from algebraic number theory to provide a quantum reduction from approx. SVP on ideal lattices (in the worst case) to random instances of the search variant of RLWE. This was followed by a novel application of automorphisms groups of cyclotomic fields for a classical search to decision reduction, thereby proving the pseudorandomness of RLWE for sets of samples polynomially bounded in the lattice dimension. In a companion publication [LPR2013b] Lyubashevsky, Peikert and Regev provided a toolkit of algorithms and techniques for applications.

The following is a description of work around the construction of key exchange methods from the ring-LWE problem. Peikert in 2009 [Pei2009b] and Lidner and Peikert [LP2011] in 2011 proposed cryptosystems and key exchange methods from both standard and RLWE assumptions. Ding [DI2012] and Ding, Xie and Lin [DXL2012] proposed optimized variants of such methods based on so called "robust extractors" that allow two parties to recover the same value from two closely separated ring elements. This was followed work by Peikert [Pei2014] who presented a key encapsulation method incorporating a low bandwidth "reconciliation" technique, which once again, allowed two parties to exactly agree on a uniformly distributed secret value from noisy RLWE instances. Subsequently
Bos, Costello, Naehrig, Stebila [BCNS2014] published a concrete instantiation with parameter sets, followed by another instantiation by Alkim, Ducas, Poppelmann and Schwabe [ADPS2015] with the same ring polynomial but a smaller modulus and a different error distribution.

2. Notation, Definitions and Operators

2.1 Algebra and Number Theory

\( Q \) : The rational numbers.

\( Z \) : The integers.

\( Z/nZ \) : Integers modulo \( n \).
   Also written as \( Z_n \).

\( R \) : The real numbers.

\( C \) : The complex numbers.

\([K:F]\) : For a field \( K \) containing \( F \),
   the degree of
   \( K/F \) as an extension of
   fields. Equivalently the
dimension of \( K \) as a vector
   space over \( F \).

\( \phi_m(x) \) : The \( m \)'th cyclotomic polynomial
   This note uses the value
   \( m = 2048 \)

\( \zeta \) : An abstract root of \( \phi_m(x) \).

\( \zeta_m \) : An \( m \)'th root of unity.

\( Q[X] \) : Polynomials with rational
coefficients

\[ Z[X] \] : Polynomials with integer coefficients

\[ K = Q(\zeta) \] : A simple extension generated over the rationals by the algebraic number \( \zeta \), the root of an irreducible polynomial over the rationals. If \( \zeta \) is an abstract root of \( \phi_m(x) \) as defined above, then \( Q(\zeta) \) is the cyclotomic field of \( m \)'th roots of unity and is isomorphic to the quotient \( Q[X]/\langle \phi_m(x) \rangle \).

Power Basis : A basis for the field \( Q(\zeta) \) when taken as a vector space over \( Q \), consisting of the elements \( \{1, \zeta , \zeta^2 ,..., \zeta^{(n-1)}\} \) where \( n = [Q(\zeta):Q] \) and \( \zeta \) is the root of an irreducible polynomial over \( Q \). In the case of the cyclotomic
field $K$, the power basis is also an integral basis for the ring of algebraic integers in $K$.

$\text{OK}$: The ring of algebraic integers of the number field $K$. As mentioned, the power basis of $K$ spans $\text{OK}$ as a $\mathbb{Z}$-module of rank $n$ and is an integral basis for the full ring of algebraic integers in $K$.

$q$: A large prime modulus such that $q = 1 \mod 2n$ where $n = [Q(\zeta):Q]$.

This note uses the value $q = (2^{31}) - 1$.

mod: The modular reduction operator

$\text{OK}_q = \text{OK}/q\text{OK}$: The quotient of $\text{OK}$ mod the ideal $q\text{OK}$.

$\text{OK}^V$: A fractional ideal, the "trace product" dual of $\text{OK}$.

$F^n$: An $n$-dimensional vector space over the field $F$. Alternatively, if $F$ is a ring, the $F$-module of
rank \( n \).

\( /x/^n \text{ in } F^n \) : Denotes a vector in \( F^n \). Written in component form as

\[ /x/^n = (x_1, x_2, \ldots, x_n). \]

\( X_\_ \) : Cartesian product, \( A \times B \) denotes the Cartesian product of \( A \) and \( B \).

\( M_\_ : OK \rightarrow C^n \) : Minkowski embedding, also called the canonical embedding, comprising injective homomorphisms from the ring \( OK \) to \( C^n \) where \( n = [Q(\zeta) : Q] \).

In this context, for two positive integers \( d_1 \) and \( d_2 \) such that \( n = d_1 + 2 \times d_2 \), \( M_\_ \) is a map from \( OK \) to the space \( R^{d_1} \times C^{(2 \times d_2)} \) iso to \( C^n \).

\( \text{Coef-Embedding} \) : Coefficient embedding. For an element \( a_\_ \) in the ring \( OK \), a representation of \( a_\_ \) in a polynomial basis of \( Q(\zeta) \).

2.2 Set Notation

\( \{,\} \) : Set brackets. \( \{a,b,c\} \), for example, denotes the set consisting of the elements
a, b and c.

\([,)\) : right-open interval. As an example, \([a,b)\) for an integer \(x\) denotes the set of integers satisfying the inequalities \(a \leq x < b\).

\(\{0,1\}^n\) : The set of \(n\) bit strings.

\(\cup\) : The union of sets.

\(A \cup B \cup C\), for example, denotes the union of sets \(A, B\) and \(C\).

\(\cap\) : The intersection of sets.

\(\in\) : Denotes set membership.

2.3 Random Variables and Distributions

\(x \leftarrow P_X()\) : Denotes the sampling of \(x\) in the set \(X\) from the probability distribution \(P_X()\) over \(X\).

\(\text{Normal}_R(,s_)\) : The (centered) 1-D Gaussian distribution over the reals with parameter \(s_\).

\(d-\text{Normal}_{Z_q}(,s_)\) : A discretization of \(\text{Normal}_R(,s_)\) over the ring \(Z_q\).

\(\text{Spherical}_R^n(,s_)\) : The (centered) spherically
symmetric n-dimensional Gaussian distribution with parameter s_ over the vector space \( \mathbb{R}^n \).

d-Spherical\(_{\mathbb{Z}_q^n}(s_\_): A\ discrete\ization\ of \ Spherical(,s_\_)_\mathbb{R}^n\ over\ the \ d-Spherical_OK_q(,s_\_)\ over\ the\ ring\ \mathbb{Z}_q^n\.

This leads to the related definition
d-Spherical_OK_q(,s_\_)\ over\ the\ ring\ \mathbb{Z}_q^n\.

An element w in \( \mathbb{Z}_q^n \) can be written as a polynomial in powers of X such that
\[
w = w_0 + w_1X + ... + w_{(n-1)}X^{(n-1)}\text{ where } n = \lfloor Q(\zeta)/Q \rfloor.
\]

To sample an element from d-Spherical_OK_q(,s_\_), it suffices to independently sample the coefficients w_0, ..., w_{(n-1)} from d-Normal\(_{\mathbb{Z}_q^n}(,s_\_).\)

Sampling an element w \in \mathbb{OK}_q
according to this method is denoted as
\[ w \leftarrow d\text{-}\text{Spherical}_{\mathbb{OK}_q}(s). \]

\text{d-Uniform}_{\mathbb{Z}_q}() : For \( x \in \mathbb{Z}_q \), the uniform distribution over \( \mathbb{Z}_q \).

\text{d-Uniform}_{\mathbb{Z}_q^n}() : The uniform distribution over the module \( (\mathbb{Z}_q)^n \).

\text{d-Uniform}_{\mathbb{OK}_q}() : (\mathbb{Z}_q)^n.

This leads to the related definition \text{d-Uniform}_{\mathbb{OK}_q}() over the ring \( \mathbb{OK}_q \).

An element \( w \) in \( \mathbb{OK}_q \) can be written as a polynomial in powers of \( X \) such that
\[ w = w_0 + w_1X + \ldots + w_{(n-1)}X^{(n-1)} \] where
\[ n = [Q(\zeta):Q]. \]

To sample an element from \( \text{d-Uniform}_{\mathbb{OK}_q}() \), it suffices to independently sample the coefficients \( w_0, \ldots, w_{(n-1)} \) from \( \text{d-Uniform}_{\mathbb{Z}_q}() \).

Sampling an element
w \in OK_q according to this method is denoted as
w <- Uniform_OK_q().

2.4 Arithmetic

*                      : a * b denotes the product of
                      a multiplied by b where a and b
                      are field or ring elements.

/                     : a / b denotes the quotient
                     of a by b.

+                     : a + b denotes the sum
                     of a and b.

-                     : a - b denotes the difference
                     of a and b.

2.5 Miscellaneous

floor(x)            : Given a real number x, outputs
                     the nearest integer less than
                     or equal to x.

nint(x)              : Given a real number x, outputs
                     floor(x + 1/2) with ties broken
                     upward.

3. Error Distribution and Embeddings

This section provides the rationale behind choice of the error
distribution, starting with Regev’s reduction from approx. GapSVP
and SIVP to LWE [Reg2005], where the error is sampled from a
discrete Gaussian. This is an approach that can be traced back to Micciancio and Regev’s [MR2004] utilization of Gaussian measures to obtain tighter bounds on the approx. SVP to SIS reduction through Gaussian sampling of offsets to random lattice points.

While the LWE error is a one dimensional Gaussian, RLWE requires that error polynomials are sampled from n-dimensional Gaussians over the Minkowski embedding of the ring \( \mathcal{O}_K \). More precisely, the literature ([LPR2013a]) defines an error distribution over the space \( \mathbb{R}^{d_1} \otimes \mathbb{C}^{(2*d_2)} \) (refer to the definition of the Minkowski embedding in section 2) as a distribution over the tensor product over \( \mathbb{Q} \) of \( \mathbb{Q}(\zeta) \) and the Reals (denoted \( \mathbb{K}_R \)). Further, in the full description of the RLWE distribution, the work exploits the fact that fractional ideals of \( \mathbb{Q}(\zeta) \) canonically embed as lattices. The RLWE secret is then drawn from a distribution over the fractional ideal \( \mathcal{O}_K \mathcal{V} \), defined as the dual of \( \mathcal{O}_K \) under a trace product. Finally, the full RLWE instance is defined as a tuple over \( (\mathcal{O}_K \mathcal{X} \otimes \mathbb{K}_R/\mathcal{Q}) \).

Certain RLWE applications, on the other hand, sample errors and secret polynomials from distributions over the ring \( \mathcal{O}_K q \) [BGV2011], [GHS2011]. In the context of power of 2 cyclotomics, this leads to a RLWE variant commonly known as polynomial LWE (PLWE Assumption - Hermite Normal Form (HNF), [BV2011]).

In a similar vein, key exchange methods [Pei2014] employ the PLWE variant by describing the ring-LWE instance as a tuple over \( (\mathcal{O}_K \mathcal{X} \otimes \mathcal{O}_K q) \) where both the secret and the error are drawn from a distribution over the ring \( \mathcal{O}_K q \) and this is the variant considered in this note.

When considering general RLWE hardness for search, the reductions require a solution for any Gaussian error distribution. The decision problem requires that Gaussian parameters are chosen at random. Despite this fact, this note employs a discretized fixed spherical Gaussian. Hardness can be established with such a distribution if the adversary is constrained to have a bounded number of samples [LPR2013a]. In practice, constructions including the key exchange method specified here, comply with this constraint.

3.1 Sampling

[BCNS2014] describe an adaptation of inverse transform sampling [Dev1986] of errors from a Gaussian. [ADPS2015] on the other hand sample errors from a centered binomial distribution partly owing to its implementation simplicity. The binomial distribution is a
suitable alternative for practical applications, however, this note follows the [BCNS2014] implementation and specifies Gaussian error. Though the [BCNS2014] implementation uses the inversion method to transform uniform variates in [0,1] to Gaussian errors, this note does not specify a sampling method and leaves it as implementation choice.

4. Functions

4.1 Modular Rounding Function [Pei2014]

For the modulus \( q \), an integer \( p \) that divides \( q \), and for for \( x \) in \( \mathbb{Z}_q \), define the modular rounding function \( \text{modR}_q_2(x) : \mathbb{Z}_q \rightarrow \mathbb{Z}_2 \) as

\[
\text{modR}_q_2(x) = \lceil (2/q) \times x \rceil \mod 2
\]

4.2 Cross Rounding Function [Pei2014]

For a modulus \( q \), and for \( x \) in \( \mathbb{Z}_q \), define the cross rounding function \( \text{crossR}_q_2(x) : \mathbb{Z}_q \rightarrow \mathbb{Z}_2 \) as

\[
\text{crossR}_q_2(x) = \lfloor (4/q) \times x \rfloor \mod 2
\]

4.3 Randomized Doubling Function [Pei2014]

Sample \( e_\text{e} \) from the set \{-1, 0, 1\} such that

\[
\Pr(0) = \frac{1}{2}, \quad \Pr(-1) = \frac{1}{4}, \quad \Pr(1) = \frac{1}{4}
\]

Where \( \Pr(x) \) is the probability of event \( x \).

Define the function \( \text{dbl}(x) : \mathbb{Z}_q \rightarrow \mathbb{Z}_{2q} \) as

\[
\text{dbl}(x) = 2x - e_\text{e}
\]

4.4 Reconciliation Function [Pei2014]

Define the following sets:

\[
I_0 = \{0, 1, \ldots, \lceil q/2 \rceil - 1\}
\]

\[
I_1 = \{-\lfloor q/2 \rfloor, \ldots, -1\} \mod q
\]
E = [-q/4,q/4) \inter Z

For an element w_ in Z_2q and b \in \{0,1\}, define the function
rec_2q(w_,b) : Z_2q X Z_2 -> Z_2 as

\{
0 if w_ \in I_b + E \mod q \\
1 otherwise
\}

4.5 Extending to the rings OK_q and OK_2q

4.5.1 Extended Modular Rounding Function

The modular rounding function, previously defined as modR_q_2(x) : Z_q -> Z_2 is extended to elements in OK_2q co-efficient wise in a polynomial basis of OK_2q. That is for an element w in OK_2q written as

w = w_0 + w_1*X + ... + w_(n-1)*X^(n-1)

where w_i \in Z_2q and n = [Q(\zeta):Q], the extended modular rounding function is defined as

modR_OK_2q_2(w) = ( modR_2q_2(w_0), modR_2q_2(w_1), ... , modR_2q_2(w_(n-1)) ) : OK_2q -> \{0,1\}^n

4.5.2 Extended Cross Rounding Function

The cross rounding function, previously defined as crossR_q_2 : Z_q -> Z_2 is extended to elements in OK_2q coefficient wise as

crossR_OK_2q_2(w) = ( crossR_2q_2(w_0), crossR_2q_2(w_1), ... , crossR_2q_2(w_(n-1)) ) : OK_2q -> \{0,1\}^n

For their arguments, both of these functions take an element in OK_2q and output an n-bit vector.

4.5.3 Extended Reconciliation Function

The reconciliation function, previously defined as rec_2q(w_,b) : Z_2q X Z_2 -> Z_2 is also extended to elements in OK_2q co-
efficient wise in a polynomial basis of $\text{OK}_2q$. For an $n$-bit vector $b_{\text{vec}}$ containing coefficient wise cross rounding information for an element in $\text{OK}_2q$:

$$b_{\text{vec}} = (b_0, b_1, \ldots, b_{(n-1)}),$$

the extended reconciliation function is defined as

$$\text{rec}_{\text{OK}_2q}(w, b_{\text{vec}}) = (\text{rec}_{2q}(w_0, b_0), \text{rec}_{2q}(w_1, b_1), \ldots, \\text{rec}_{2q}(w_{(n-1)}, b_{(n-1)})) : (\text{OK}_2q \times \{0,1\}^n) \rightarrow \{0,1\}^n$$

For its arguments, this function takes an element $w$ in $\text{OK}_2q$ and the $n$-bit vector containing co-efficient wise cross rounding information for $w$ and returns an $n$-bit vector.

### 4.5.4 Extended Randomized Doubling Function

Finally, the randomized doubling function, previously defined as $\text{dbl}(x) : \mathbb{Z}_q \rightarrow \mathbb{Z}_{2q}$ is also extended to elements in $\text{OK}_q$ co-efficient wise in a polynomial basis of $\text{OK}_q$. For an element in $\text{OK}_q$ defined as

$$w = w_0 + w_1X + \ldots + w_{(n-1)}X^{(n-1)}$$

where $w_i \in \mathbb{Z}_{2q}$ and $n = [Q(\zeta) : Q]$, the doubling function $\text{dbl}_{\text{OK}_q}(w)$ is defined as

$$\text{dbl}_{\text{OK}_q}(w) = \text{dbl}(w_0) + \text{dbl}(w_1)X \ldots + \\text{dbl}(w_{(n-1)})X^{(n-1)} : \text{OK}_q \rightarrow \text{OK}_{2q}$$

For its arguments, the extended doubling function takes an element in $\text{OK}_q$ and returns an element in $\text{OK}_{2q}$. 
5. Flow Diagram

Server

------

Client

\[ a \leftarrow d\text{-Uniform}_q() \]

\[ s_1, e_1 \leftarrow d\text{-Spherical}_q(s) \]

\[ s_2, e_2, e_3 \leftarrow d\text{-Spherical}_q(s) \]

\[ b = a \cdot s_1 + e_1 \]

\[ u = a \cdot s_2 + e_2 \]

\[ v = b \cdot s_2 + e_3 \]

\[ v_\cdot = \text{dbl}(v) \]

\[ v^* = \text{crossR}_2q_2(v_\cdot) \]

\[ u, v^* \leftarrow \]

\[ \text{pms} = \text{rec}_2q(2u \cdot s_1, v^*) \]

\[ \text{ms} = \text{KDF}(	ext{pms}) \]

\[ \text{pms} = \text{modR}_2q_2(v_\cdot) \]

\[ \text{ms} = \text{KDF}(	ext{pms}) \]

\[ a, b, s_1, s_2, e_2, e_3, u \text{ and } v \text{ are elements in } \text{OK}_q, \]

\[ v_\cdot \text{ is an element in } \text{OK}_q \text{, } v^* \text{ and } \text{pms} \text{ (the pre-master secret) are both } \]

1024 bit vectors, \[ \text{ms} \text{ is the master secret. } \text{KDF}() \text{ is a key} \]

\[ \text{derivation function. In practice, it is expected that the key} \]

\[ \text{derivation method specified in TLS 1.2 section 8.1} \] [RFC5426] \[ \text{will} \]

\[ \text{be used (refer to subsequent section on TLS extensions). For an} \]

\[ \text{alternate approach to generation of the parameter } \text{a'} \] [ADPS2015] \[ \text{refer the security section of this note.} \]

Figure 1: Flow Diagram

6. Efficient Polynomial Operations

This note does not specify methods for efficient polynomial
operations, this is a choice that is left to for implementations to consider. The Number Theoretic Transform (NTT) has been used as a technique for fast polynomial multiplication in rings of algebraic integers ([PG2012], [PG2013], [ADPS2015]), and is therefore particularly well suited for lattice based cryptography.

7. Protocol

7.1 Server

7.1.1 Ring Element ‘a’

Server generates the ring element a by sampling from the uniform distribution over OK_q

\[ a \leftarrow d-\text{Uniform}_{OK_q}() \]

This can be done by sampling each coefficient of a independently from the uniform distribution over Z_q

7.1.2 Server Secret Key and Error

Server generates its ephemeral secret key s_1 and the error e_1 sampling the ring elements from the discrete spherical Gaussian distribution over OK_q with parameter s_.

\[ s_1 \leftarrow d-\text{Spherical}_{OK_q}(,s_1) \]
\[ e_1 \leftarrow d-\text{Spherical}_{OK_q}(,s_1) \]

This can be done by sampling each coefficient of s_1 and e_1 independently from the discrete Gaussian distribution over Z_q centered at 0 with parameter s_.

7.1.3 Server Public Key

Server computes its ephemeral public key b by multiplying the element a and its ephemeral secret key s_1 and then adding the error e_1
\[ b = a \cdot b + e_1 \]

7.1.4 Server Key and Parameter Exchange

Server returns the ring element \( a \) and the its ephemeral public key \( b \) to the client.

7.1.5 Client Key and Cross Rounding Vector Receipt

Server receives the client public key \( u \) along with cross rounding information \( v^* \) from the client.

7.1.5 Server Reconciliation and Key Derivation

The server then computes \( h = 2 \cdot u \cdot s_1 \) by multiplying the ephemeral client public key with its ephemeral secret key and doubling the result. The server then computes the premaster secret (pms) as:

\[ \text{pms} = \text{rec}_{\text{OK}_2q}(h, v^*) \]

An approved key derivation function can then be used for deriving the master secret key. In practice, it is expected that the key derivation method specified in TLS 1.2 [RFC5426] section 8.1 will be used.

7.2 Client

7.2.1 Client Secret Key and Error

The client generates its ephemeral secret key \( s_2 \) and the error terms \( e_2 \) and \( e_3 \) by sampling the ring elements from the discrete spherical Gaussian distribution over \( \text{OK}_q \) with parameter \( s_\).

\[ s_2 \leftarrow d\text{-Spherical}_{\text{OK}_q}(s) \]

\[ e_2 \leftarrow d\text{-Spherical}_{\text{OK}_q}(s) \]

\[ e_3 \leftarrow d\text{-Spherical}_{\text{OK}_q}(s) \]

This can be done by sampling each coefficient of \( s_2 \) and \( e_2 \) and \( e_3 \) independently from the discrete Gaussian distribution over
7.2.2 Server Key and Parameter Receipt

The client receives the parameter $a$ from the server and the ephemeral server public key $b$.

7.2.3 Client Public Key

The client computes its ephemeral public key $u$ by multiplying elements $a$ and its ephemeral secret key $s_2$ and then adding the error $e_2$.

$$u = a \cdot s_2 + e_2$$

7.2.4 Client Doubling and Cross Rounding

Client computes the element $v$ by multiplying the ephemeral server public key $b$ with its secret key $s_2$ and adding the error term $e_3$.

$$v = b \cdot s_2 + e_3$$

The client then applies the extended randomized doubling function to $v$ and obtains the element $v_\_ in OK_{2q}$:

$$v_\_ = \text{dbl}_{OK_q}(v)$$

The client then computes the cross rounding of $v_\_$ by applying the extended cross rounding function to $v_\_$ and obtains the n-bit vector $v^*$

$$v^* = \text{CrossRound}_{OK_{2q}}(v_\_)$$

7.2.5 Client Key and Cross Rounding Vector Exchange

The client returns its ephemeral public key $u$ and the cross rounding vector $v^*$ to the server.

7.2.6 Client Modular Rounding and Key Derivation

The client then applies the extended modular rounding function to $v_\_$ and computes the premaster secret (pms) as:
\[ pms = \text{modR\_OK\_2q\_2}(v) \]

An approved key derivation function can then be used for deriving the master secret key. In practice, it's expected that the key derivation method specified in TLS 1.2 section 8.1 [RFC5426] will be used.

8. TLS Extensions

This section addresses authentication as it relates to key exchange along with guidance for integration of this key exchange method with the TLS protocol.

Authenticated key exchange (AKE) protocols allow two parties to mutually establish ephemeral authenticated session keys for two way communications. There is a body of literature in the area of formalizing models for AKE, including the significant work of Canetti and Krawczyk [CN2002] on demonstrating security of so-called "post-specified peer" SIGMA ("SIGn-and-MAC") protocols in the simulation paradigm. Whereas protocols such as IKE achieve some of these goals, the passive lattice based key exchange protocol described here does not. The approach to address this deficiency is to integrate it within the TLS handshake protocol. There are two ways to do this, the first as a "fundamental" replacement for FFC/ECC discrete log based classical key exchange methods [NIST-SP-800-56A] which will be described below, and the second being a "hybrid" approach that incorporates this lattice based method with discrete log based key exchange (which is clearly the more conservative approach that should be considered by implementations).
8.1 Fundamental RLWE Suites

This note therefore follows after the guidance in [BCNS14] and its associated implementation for TLS versions supporting AEAD modes by specifying the following cipher suites. Also, this note follows the blueprint for specifying TLS extensions set forth in [RFC4492] (which defines the TLS extensions for Elliptic Curve cipher suites):

```
+-----------------------------------------+
|                              |
|      TLS_RLWE_RSA_AES128_GCM_SHA256    |
|                              |
|      TLS_RLWE_ECDSA_AES128_GCM_SHA256 |
|                              |
+-----------------------------------------+
```

Figure 3: Fundamental RLWE Cipher Suites

8.2 Fundamental RLWE Key Exchange Algorithms
The following server key exchange algorithms are specified

8.2.1 RLWE_RSA

For this method the server’s certificate must contain a RSA capable signing key and be signed with RSA. The server sends its ephemeral RLWE public key, the ring element ‘\(a\)’ and a specification of the lattice parameters in the ServerKeyExchange message. These parameters MUST be signed with RSA using the private key corresponding to the public key in the server’s Certificate. The client generates an ephemeral RLWE public and secret keys based on the server’s lattice parameters and ring element ‘\(a\)’ its ephemeral public key and the cross rounding vector in the ClientKeyExchange message. The client then performs the modular rounding operation whereas the server performs the reconciliation operation (as described in the protocol section above). Client and server then use the resultant shared secret as the premaster secret.

8.2.2 RLWE_ECDSA

This key exchange algorithm is the same as ECDH_RSA except that the server’s certificate MUST be signed with ECDSA rather than RSA.

8.3 Fundamental TLS extensions for RLWE

A single new TLS extension is specified in this section, called the Fundamental RLWE Parameter Set extension (which specifies the ring polynomial, the modulus \(q\), the distribution type, standard deviation and expected value, see section immediately below for more details).

Though a single set of parameters is described here, in the future these extensions should allow for negotiating other lattice parameters. The client should enumerate the lattice parameters it supports in its ClientHello message. The server should in a similar way enumerate the lattice parameters it supports in its ServerHello message. A TLS client that proposes RLWE cipher suites in its ClientHello message SHOULD include this extension. Servers implementing RLWE cipher suites MUST support this extension, and when a client uses this extension, servers MUST NOT negotiate the use of a RLWE cipher suite unless they can complete the handshake under the choice of RLWE parameters supported by the client. The client MUST NOT include this
extension in the ClientHello message if it does not propose any RLWE cipher suites.

8.3.1 Fundamental RLWE Parameters Extension

text
enum {
    sec_cyclotomic_2048 (1), reserved (..)
} RingPolynomial;

text
enum {
    sec_mod_2_32_1 (1), reserved (..)
} Modulus;

text
enum {
    sec_gaussian (1), reserved (..)
} Distribution;

text
enum {
    sec_std_8_Sqrt_2_by_Pi (1), reserved (..)
} StandardDeviation;

text
enum {
    sec_ev_0 (1), reserved (..)
} ExpectedValue;

text
struct {
    RingPolynomial poly;
    Modulus mod;
    Distribution dist;
    StandardDeviation std;
    ExpectedValue ev;
} FundamentalRLWEParamSet;

struct { FundamentalRLWEParamSet paramSetList<1..2^8-1>;
} FundamentalRLWEParamSetList;

In this version of the note, the only valid instantiation of the struct FundamentalRLWEParamSet is the following:

struct FundamentalRLWEParamSet params =

{ sec_cyclotomic_2048, sec_mod_2_32_1, sec_gaussian, \
sec_std_8_Sqrt_2_by_Pi, sec_ev_0 };

Khera, Rohit          Expires April 24 2018         [Page 24]
Implementations MUST ensure that passed parameters agree with the above. The above namespace will need to be maintained by IANA.

8.3.2 Client Hello Extension

This section specifies the TLS extension that can be included with the ClientHello message for RLWE based key exchange. This extension is the Fundamental RLWE Parameter Set extension.

When this extension is sent:

The extension SHOULD be sent along with any ClientHello message that proposes RLWE cipher suites. The extension allows a client to enumerate the RLWE parameter sets that it supports. The general structure of TLS extensions is described in [RFC4366], and this note specification adds the following to ExtensionType.

enum { FundamentalRLWE(..) } ExtensionType;

FundamentalRLWE (Fundamental RLWE Parameter Set extension):

Indicates the Fundamental RLWE parameters supported by the client. For this extension, the opaque extension_data field contains the FundamentalRLWEParmSetList. See the preceding section for details.

Actions of the sender:

A client that proposes RLWE cipher suites in its ClientHello message appends this extension. In the current version of this note, clients SHOULD only send the single parameter set outlined in the preceding section.

Actions of the receiver:

A server that receives a ClientHello containing this extension MUST use the client’s enumerated capabilities to guide its selection of an appropriate cipher suite. One of the proposed RLWE cipher suites must be negotiated only if the server can successfully complete the handshake while using the Fundamental RLWE parameter set supported by the client.

8.3.3 Server Hello Extension

This section specifies a TLS extension that can be included with the ServerHello message for RLWE based key exchange (the
When this extension is sent:

The Fundamental RLWE Parameter Set Extension is included in a ServerHello message in response to a ClientHello message containing the Fundamental RLWE Parameter Set Extension when negotiating an RLWE cipher suite.

Meaning of this extension:

This extension allows a server to enumerate RLWE parameters that it supports for the parameter and key material exchanges in its ServerKeyExchange message.

Structure of this extension:

The server’s Fundamental RLWE Parameter Set Extension has the same structure as the client’s Fundamental RLWE Parameter Set Extension (see preceding section).

Actions of the sender:

A server that selects a RLWE cipher suite in response to a ClientHello message (that includes a Fundamental RLWE Parameter Set) and appends this extension to its ServerHello message.

Actions of the receiver:

A client that receives a ServerHello message containing a Fundamental RLWE Parameter Set Extension MUST respect the server’s choice of RLWE parameters.

8.3.4 Server Certificate

This message is sent for all non-anonymous Fundamental RLWE key exchange algorithms. No additional or modified processing is required for this message for the fundamental RLWE key exchange methods described here.

8.3.5 Server Key Exchange

When this message is sent:

This message is sent when using the RLWE_ECDSA and RLWE_RSA key
exchange algorithms. This message is used to convey the (i) the fundamental RLWE parameter set (ii) the public ring element ‘a’ (refer to preceding sections on protocol flow and description), along with the server’s ephemeral public key b.

struct {
    FundamentalRLWEParamSet params;
    opaque a <2^12>;
    opaque b <2^12>
} ServerFundamentalRLWEParams;

The ServerKeyExchange message is extended as follows:

enum { fundamental_RLWE_kex } KeyExchangeAlgorithm;

This indicates the ServerKeyExchange message contains the ring element ‘a’ and the ephemeral server RLWE public key.

select (KeyExchangeAlgorithm) {
    case fundamental_RLWE_kex:
        ServerFundamentalRLWEParams params;
        Signature signed_params;
    } ServerKeyExchange;

params: Specifies the server ephemeral RLWE public key, the ring element ‘a’ and associated fundamental RLWE parameters

signed_params: Consists of a hash of the params, along with the appropriate signature corresponding to the key exchange method (i.e. RLWE_RSA or RLWE_ECDSA). The private key corresponding to the certified public key in the server’s Certificate message is used for signing.

Actions of the sender:

The server selects the Fundamental RLWE parameters (see prior section for a description of valid parameters), ephemeral public key and the ring element ‘a’ and conveys this information to the client in the ServerKeyExchange message using the format defined above.

Actions of the receiver:

The client verifies the signature and retrieves the server’s RLWE parameters, ephemeral public key and the ring element ‘a’ from the ServerKeyExchange message.
8.3.6 Certificate Request

This message sent when requesting client authentication. No additional or modified processing is required for this message for the Fundamental RLWE key exchange methods described here.

8.3.7 Client Certificate

This message is sent in response to a CertificateRequest. No additional or modified processing is required for this message for the fundamental RLWE key exchange methods described here.

8.3.8 Client Key Exchange

This message contains the client’s ephemeral RLWE public key and cross rounding information.

Meaning of the message: This message is used to convey ephemeral data relating to the key exchange belonging to the client.

Structure of this message: The TLS ClientKeyExchange message is extended as follows.

```c
struct {
    FundamentalRLWEParamSet params;
    opaque u <2^12>;
    opaque cr <2^7>;
} ClientFundamentalRLWEParams;
```

This message contains the client’s ephemeral RLWE public key and cross rounding information.

```c
struct {
    select (KeyExchangeAlgorithm) {
        case fundamental_RLWE_kex:
            ClientFundamentalRLWEParams;
    } exchange_keys;
} ClientKeyExchange;
```

Actions of the sender:

The client generates its ephemeral public key upon receipt of the parameter ‘a’ from the server and sends this along with the cross rounding vector to the server.

Actions of the receiver:
The server retrieves the client’s ephemeral RLWE public key from the ClientKeyExchange message and ensures that it is consistent with the agreed upon Fundamental RLWE parameters.

8.3.9 Certificate Verify

Pending

8.4 Hybrid RLWE Suites

This section, which is pending, will specify TLS extensions for the use of RLWE key exchange in conjunction with FFC/ECC discrete log based key exchange.

The hybrid method will be the preferred key exchange method specified in this note.

+----------------------------------------------------+
|                                                     |
| TLS_RLWE_DHE_RSA_AES128_GCM_SHA256                 |
| TLS_RLWE_ECDHE_RSA_AES128_GCM_SHA256              |
| TLS_RLWE_DHE_ECDSA_AES128_GCM_SHA256             |
| TLS_RLWE_ECDHE_ECDSA_AES128_GCM_SHA256           |
+----------------------------------------------------+

Figure 4: Hybrid RLWE Cipher Suites

9. Authenticity

As discussed in the previous section on TLS integration, the proposed approach relies on classical signature schemes RSA and ECDSA [FIPS186-4] for authenticity. For compatibility with 128 bit security, implementations should use 3072 RSA and ECDSA on the nist curve p256.

10. Code Implementations and Licensing

The following code is the closest code implementation of the method described in this note and is attributed to [BCNS14]:

Khera, Rohit                 Expires April 24 2018                 [Page 29]
https://github.com/dstebila/rlwekex The licensing terms are from: http://unlicense.org

An OpenSSL fork incorporating RLWE key exchange is available here under the terms of the OpenSSL license: https://github.com/dstebila/openssl-rlwekex/tree/OpenSSL_1_0_1-stable

The availability of [ADPS2015] implementation is also noted. This implementation utilizes a smaller modulus and a different reconciliation method, and is available under a public license from: https://github.com/tpoepelmann/newhope and from: https://cryptojedi.org/crypto/#newhope

11. IANA Considerations

IANA is required to maintain the namespaces for the following TLS extensions:

1) FundamentalRLWEParamSet
2) ServerFundamentalRLWEParams
3) ClientFundamentalRLWEParams
4) Pending - namespaces for hybrid key exchange

12. Security Considerations

The security of the key exchange method described here stems from a quantum reduction from approx. SVP on ideal lattices in the worst case to random instances of the search RLWE problem. The pseudorandomness of RLWE has been proven through a classical search to decision reduction for Galois number fields. The reader is referred to the introduction of this document as well as the section on error distributions for more details and for citations to the relevant literature.

The security of the RLWE instantiations depend on the choice of error distribution. In particular, security with a fixed spherical error has been established for bounded numbers of RLWE samples, a constraint that is achieved in the key exchange method described here. The reader is referred to the introduction and the section on error distributions for more details.

Incorrect choice of parameters and errors can lead to vulnerable
RLWE instantiations. Rather than state the nature and cryptanalysis of such instantiations, the reader is referred to the literature on this subject such as [Pei2016a].

This key exchange method relies on the public ring element a_ (refer to description of the high level flow). Rather than specify this to be a global, fixed parameter, this note takes the approach of sampling this element uniformly from the ring OK/qOK for every key exchange.

12.1 Security Parameters

In order to fully describe the RLWE distribution used in this key exchange method, the ring polynomial, modulus "q" and the 1-D Gaussian standard deviation and expected value need to be defined. In order to do so, this note adopts the following parameter sets from [BCNS2014]:

```
+---------------------------------------------+
|                                             |
|  Ring polynomial        : 2048th cyclotomic  |
|  Modulus q              : (2^31) -1         |
|  Error distribution     : Gaussian          |
|  Guassian parameter (s_): 8 / Sqrt(2*Pi)    |
|  Expected value         : 0                 |
+---------------------------------------------+
```

Figure 5: Parameter Sets

These parameters provide a security of 128 bits against classical attacks as described in the literature ([LP2011],[BCNS2014]).

12.2 Ring Element ‘a’

This key exchange method relies on the public ring element a_ (refer to description of the high level flow). Rather than specify this to be a global, fixed parameter, in this note, the server samples this element uniformly from the ring OK/qOK for every key exchange and passes this element to the client (in the TLS setting this is done in the Server Key Exchange message). It is noted that [ADPS2015] on the other hand use the approach where
one of the parties samples a seed and generates a by application of the SHAKE-128 function. This is an approach that may be considered in a future version of this draft.

12.3 Side Channels

This note treats certain computational aspects such as efficient polynomial multiplication as an implementation choice. In a similar vein, it is expected that implementations make their own determination around countermeasures against side channel attacks.

13. References

13.1 Normative References


[NIST-SP-800-56A] NIST Special Publication 800-56A Revision 2, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"


[RFC4506] Eisler, M., "XDR: External Data Representation"
13.2 Informative References


[DI2012] Jintai Ding, "A simple provably secure key exchange scheme
based on the learning with errors problem”. IACR Cryptology ePrint
Archive report 2012/688.

[DXL2012] Jintai Ding, Xiang Xie, Xiaodong Lin, "A Simple Provably
Secure Key Exchange Scheme Based on the Learning with Errors

[GHS2011] Craig Gentry and Shai Halevi and Nigel P. Smart, "Fully
Homomorphic Encryption with Polylog Overhead", Cryptology ePrint
Archive, Report 2011/566

[LP2011] "Lindner, Richard and Peikert, Chris", "Better Key Sizes
(and Attacks) for LWE-Based Encryption", In proceedings "Topics in
Cryptology -- CT-RSA 2011: The Cryptographers’ Track" at the RSA
Heidelberg, pgs 319--339

[LPR2013a] Lyubashevsky, Vadim and Peikert, Chris and Regev, Oded,
"On Ideal Lattices and Learning with Errors over Rings". J. ACM,
November 2013 Vol.60/6, 2013.

[LPR2013b] Vadim Lyubashevsky and Chris Peikert and Oded Regev, "A
Toolkit for Ring-LWE Cryptography". Cryptology ePrint Archive,

[MR2004] Daniele Micciancio, Oded Regev, "Worst-Case to Average-
Case Reductions Based on Gaussian Measures", vol. 00, pp. 372-381,
2004, FOCS.2004.72

[Pei2009a] Chris Peikert, "Public-key cryptosystems from the worst-
case shortest vector problem". In STOC, pages 333-342. 2009

[Pei2009b] Chris Peikert, "Public-key cryptosystems from the worst-
case shortest vector problem, 2009".
https://web.eecs.umich.edu/~cpeikert/pubs/svpcrypto.pdf

[Pei2014] Chris Peikert} "Lattice Cryptography for the Internet",
Cryptology ePrint Archive Report 2014/070, 2014


14. Acknowledgements

The author would like to thank Pivotal for support in putting together this draft. The author would also like to thank Eric Malm for insightful discussions and perspectives into the relevant aspects of algebra and algebraic number theory.

15. Author’s Address

Rohit Khera
Pivotal
875 Howard St.
San Francisco
CA 94103

EMail: rkhera@pivotal.io