I. J. Computer Network and Information Security, **2016, 4, 44-55**
Published Online April 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2016.04.06

Formal Verification of NTRUEncrypt Scheme

Gholam Reza Moghissi
ICT Department, Malek-e-Ashtar University of Technology, Tehran, Iran
E-mail: ftmsecure_ubbao@mut.ac.ir

Ali Payandeh
ICT Department, Malek-e-Ashtar University of Technology, Tehran, Iran
E-mail: payandeh@mut.ac.ir

Abstract—In this paper we explore a mechanized verification of the NTRUEncrypt scheme, with the formal proof system Isabelle/HOL. More precisely, the functional correctness of this algorithm, in its reduced form, is formally verified with computer support. We show that this scheme is correct what is a necessary condition for the usefulness of any cryptographic encryption scheme. Besides, we present a convenient and application specific formalization of the NTRUEncrypt scheme in the Isabelle/HOL system that can be used in further study around the functional and security analysis of NTRUEncrypt family.

Index Terms—Public key encryption, Lattice, NTRUEncrypt, Formal proof system, Higher order logic, Formal verification, Isabelle/HOL, Theorem proving, Proof assistant.

I. Introduction

Formal proof systems are a useful approach in the area of verification. Formal and computer verification augment the traditional concept of software engineering by providing techniques that guarantee trustiness as well as correctness of software systems in a mathematical way. There are many possible applications of formal verification like automotive, medical technology, information technology (software/hardware security, network, cryptography and protocol) and so on. Some of the researches around with formal verification of cryptographic functions, such as *Formal Analysis of a Public-Key Algorithm* (functional verification and computational security of rabbin encryption and digital signature scheme)[4], *Formal Proof for the Correctness of RSA-PSS* (functional verification of probabilistic signature scheme RSA-PSS)[20], *A Computer Proven Application of the Discrete Logarithm Problem* [6](from the thesis: *Formalizing the DSA Signature Scheme in Isabelle/HOL*[5]), *Verification of Cryptographic Primitive SHA-256* [23] and so on, encourage us to formalize and verify the NTRUEncrypt Scheme.

Now a days, because of the challenge of powerful quantum computer, one of the important scheme in the public-key cryptography is NTRUEncrypt function as suitable post quantum encryption scheme[16]. NTRUEncrypt public-key scheme (not fully) relies on the hardness of solving SVP (Shortest Vector Problem which is the problem of finding shortest vector $\hat{t}$ in the lattice of $\mathcal{L}$ as $\hat{t} \in \mathcal{L}$) and CVP (Closest Vector Problem which is the problem of finding a vector $\hat{t} \in \mathcal{L}$ that is closest to a target vector $t \in R^n$ not belong to lattice $\mathcal{L}$) in a convolution modular lattice (since security analysis of NTRUEncrypt Scheme not included in this research we don't open them). In this paper, we explore a computer verification for simplified version of the NTRUEncrypt scheme with the formal proof system Isabelle/HOL. More precisely, the functional correctness of this algorithm is formally verified in a simplified version with computer support. Besides, we present a convenient formalization of the NTRUEncrypt scheme in the Isabelle/HOL system that can be used as a framework for further studies on this function. Further Studies in this domain can be followed by the topics such as: computer verification of security properties using a straight-forward computation model in Isabelle/HOL, applying of Isabelle/HOL to the complete version of NTRUEncrypt function, checking the applicability of deferent introduced traditional attacks on the NTRUEncrypt scheme, checking the applicability of quantum attacks on the NTRUEncrypt scheme (which need to formalize the necessary quantum primitives and theorems in Isabelle/HOL), and so on.

The NTRUEncrypt public key cryptosystem, also known as the NTRU encryption algorithm[1], is a lattice-based alternative to RSA and ECC which is based on the shortest vector problem (SVP) in a lattice (which is known not to be breakable using quantum computers). The first version of the system, which was simply called NTRU, was developed around 1996 by three mathematicians (J. Hoffstein, J.Pipher and J.H. Silverman)[1]. In 1996 these mathematicians together with D. Lieman founded the NTRU Cryptosystems, Inc. and were given a patent on the cryptosystem. Now the system is fully accepted to IEEE P1363 standards under the specifications for lattice-based public-key cryptography (IEEE P1363.1)[12, 16]. Because of the speed of NTRUEncrypt public key cryptosystem and its low memory use, it can be used in applications such as mobile devices, Smart-cards and so on. In April 2011, NTRUEncrypt was accepted as a X9.98 Standard, for use in the financial services industry[16].

This paper is organized as follows: We start in Chapter 2 with a description of the used formal proof system. In

Chapter 3 we explore the mathematical background of NTRUEncrypt scheme. In Chapter 4 we present a main formalization of the NTRUEncrypt Algorithm, and in chapter 5 we introduced the issues related to verification of the NTRUEncrypt Algorithm (which include some formalization of scheme). At the end, in Chapter 5 some conclusions, as well as some comments on future works are given.

II. Isabelle Formal Proof System

Isabelle is a generic system for implementing logical formalisms which invented at Cambridge University and TU Munich[9]. Isabelle/HOL is the specialization of Isabelle for HOL (which abbreviates Higher-Order Logic), and can be applied to several logics[9]. Other logics distributed with Isabelle include the usual first-order logic (FOL) or LCF (which is a version of Scotts logic for computable functions). Isabelle/HOL allows to turn executable specifications directly into codes in SML, OCaml, Haskell, and Scala[22].

Isabelle is also often referred to as a “Proof Assistant” underlining the process of alternating automated reasoning with human intervention. Theorem proving with Isabelle is often based on a “human-guided” manipulation of the proof state, where the system itself only executes the given commands and verifies their applicability until finally all subgoals have been proven (e.g. via reduction to already proven lemma). On the other hand there are also strong tools that can be applied to handle (suitable) proofs (or at least considerable parts of it) automatically.

Isabelle comes with a large theory library of formally verified mathematics, including elementary number theory, analysis, algebra, set theory and so on. Using appropriate *include* commands we may base our (new) theorems upon those in the libraries by referring to them during the proof process wherever they are applicable. More informations about the Isabelle/HOL are given in [7-10, 22].

III. NTRU Encryption Scheme

In 1998 Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman introduced NTRU, a new public key cryptosystem in his well known paper[1], *"NTRU: A Ring-Based Public Key Cryptosystem"*. The original idea for NTRUEncrypt is due to Hoffstein in 1994 and the system was developed by Hoffstein, Pipher, and Silverman during 1994-1996. NTRU features reasonably short key, easily created keys, high speed functions, and low memory requirements.

The encryption procedure of NTRU cryptosystem uses a mixing system based on polynomial algebra and reduction modulo two numbers p and q, while the decryption procedure uses an unmixing system which its validity depends on elementary probability theory[1]. The security of the NTRU public key cryptosystem comes from the interaction of the polynomial mixing system with the independence of reduction modulo p and q. NTRUEncrypt is in fact a lattice-based public key cryptosystem, because underlying convolution polynomial ring $\mathbb{Z}[X]/(X^N - 1)\ mod\ q$ is Convolution Modular Lattice and security of it also rests on the difficulty of solving CVP and SVP in these lattices.

NTRUEncrypt is a probabilistic cryptosystem, meaning that encryption includes a random element, so each message has many possible encryptions. Also, decryption Process may cause failure such that you can estimate this decryption failure for different centering algorithm (described in subsection 3.4) [3]. It is showed that for a recommended parameter sets, the chance of decryption failure can be less than 2^{-100}!

A. NTRU Encryption Notation

NTRU cryptosystem depends on three integer parameters (n, p, q) and four set of integer[1] vector with length n (or polynomials of degree $n-1$ with integer coefficients): $\mathcal{L}_\mathrm{f}$, $\mathcal{L}_\mathrm{g}$, $\mathcal{L}_r$, $\mathcal{L}_\mathrm{m}$. Vectors belong to each of these four sets, are short (in practice with the entries of $+1$, -1 and 0), such that have a specific Integer bound[2]: $\mathcal{L}_\mathrm{f} = \mathcal{L}(d_f, d_f - 1)$, $\mathcal{L}_\mathrm{g} = \mathcal{L}(d_g, d_g)$, $\mathcal{L}_\mathrm{r} = \mathcal{L}(d_r, d_r)$ and $\mathcal{L}_\mathrm{m} = \mathcal{L}(d_m, d_m)$. We assume that $gcd(p,q) = 1$, $q \gg p$ and $n > 0$. The vector f have an inverse in modulo q (shown by Fq) and have an inverse in modulo p (shown by Fp). A vector or polynomial $F \in \hat{R}$ in the ring $\hat{R} = [X]/(X^n - 1)$ will be written as[1]:

$$F = \sum_{i=0}^{n-1} F_i x^i = [F_0, F_1, \dots, F_{n-1}] \quad (1)$$

We write $\otimes$ to denote multiplication in $\hat{R}$ as a cyclic convolution product in $O(n^2)$ operations:

$$F \otimes G = H\ with \quad (2)$$

$$H_k = \sum_{i=0}^{k} F_i G_{k-i} + \sum_{i=k+1}^{n-1} F_i G_{n+k-1} = \sum_{i+j \equiv k (mod\ n)} F_i G_i$$

The ring $\hat{R}$ allow us to compute $F \otimes G = H$ as[2]:

$$\begin{bmatrix} f_1 & f_n & \cdots & f_3 & f_2 \\ f_2 & f_1 & \cdots & f_4 & f_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & \cdots & f_1 & f_n \\ f_n & f_{n-1} & \cdots & f_2 & f_1 \end{bmatrix} * \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_{n-1} \\ h_n \end{bmatrix} \quad (3)$$

In this paper, multiplication of an integer number with a vector showed by "$\cdot$", multiplication of an integer number with an integer number showed by "$*$" and addition of two vectors showed by "$\oplus$".

The width of a polynomial $A(x)$ is the difference between its largest coefficients, $Max(A(x)) =$

[1] It is arbitrary to select every rings in modulo of an integer number

[2] The set of $\mathcal{L}(d_{+1}, d_{-1})$ contains the vectors that have d_{+1}coefficients of $+1$ and d_{-1}coefficients of -1

$\max\{a_0, a_1, \ldots, a_{n-1}\}$ and smallest coefficients, $Min\big(A(x)\big) = \min\{a_0, a_1, \ldots, a_{n-1}\}$ which is shown as:

$$Width\big(A(x)\big) = Max\big(A(x)\big) - Min\big(A(x)\big) \qquad (4)$$

B. Key Creation

To create an NTRU keys, two randomly polynomials f and g should be selected from $\mathcal{L}_\mathrm{f}$ and $\mathcal{L}_\mathrm{g}$, that f have inverse in modulo q and modulo p [1]. We will denote these inverses by F_q and F_p, that is, $F_q \otimes f \equiv 1\ (mod\ q)$ and $F_p \otimes f \equiv 1\ (mod\ p)$, next compute the public key $h \equiv F_q \otimes g\ (mod\ q)$ and private key of the system is (f, F_p).

C. Encryption Process

The plaintext M and additional random bits R are used to select a pair of encoded plaintext polynomials $(r, m) \in \mathcal{L}_\mathrm{r} \times \mathcal{L}_\mathrm{m}$ according to a public encoding scheme $\mathcal{E}$. The knowledge of (r, m) allows easy recovery of M and R in practice (also, knowledge of m alone allows easy recovery of M and R, from which r may be recomputed using $\mathcal{E}$)[3]: $\mathcal{E}(M, R) = (m, r)$ and $\mathcal{E}_{Inv}([], m) = (R, M)$. To Encrypt the m selected from $\mathcal{L}_\mathrm{m}$, with randomly chosen polynomial r from $\mathcal{L}_\mathrm{r}$ and using the public key h, we should compute $e \equiv p \cdot r \otimes h \oplus m\ (mod\ q)$[1].

D. Decryption Process

To decrypt the encrypted message e by private key, we should perform the following steps[1]:

1) $a \equiv f \otimes e$
2) $if(Width(a) < q)\ then\ gap_{failure} = false$
3) $if\big(gap_{failure}\big) then\ Centring_{Start}$
4) $Centring_{Start}: a = \mathcal{R}(a)$
5) $m \equiv F_p \otimes a\ (mod\ p)$
6) $(R, M) = \mathcal{E}_{Inv}([], m)$

Because the polynomials r, g, f, m are small, their products will in general have low width, thus we can find a correct interval that can reduce a in it so that $\mathcal{R}(a) = \mathcal{R}(a)\ (mod\ q)$ (formula $\mathcal{R}(a)$ refers to Reduced a in the correct interval) and decryption algorithm exactly will recover the m[3, 21]. if we have selected wrong interval, the recovered value will defer from m. A decryption failure will occur if $Width(p \cdot r \otimes g \oplus f \otimes m) \geq q$ (referred to *gap failure*) or if $Width(p \cdot r \otimes g \oplus f \otimes m) < q$ but we have reduced into wrong interval (referred to *wrap failure*). If gap failure was not observed, we should apply centring method on a to compute $\mathcal{R}(a)$ that satisfy the following condition $\mathcal{R}\ (a)\ mod\ q = p \cdot r \otimes g \oplus f \otimes m$ (two centring method introduced in [3]).

One of the important problem in NTRUEncrypt is the parameter selection to have lowest Decryption failure with most efficient speed and storage. The parameter selection will not discussed here (the same as centering method), we should be able to compute the probability of decryption failure[3]. The likelihood of a decryption failure can be made arbitrarily small, IEEE P1363.1 says in appendix A.4.10[12] that for ternary polynomials with d coefficients of $+1$ and the same number of -1, the chance of a decryption failure is given by[13]:

$$\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)} = P_{(d,n)}\left(\frac{q-2}{6}\right) \qquad (5)$$

where

$$P_{(d,n)}(c) = n \times erfc\left(\frac{c}{\sigma\sqrt{2n}}\right)\ and\ \sigma(d,n) = \sqrt{\frac{8d}{3n}}$$

where $erfc$ is the Gauss error function (as a practical example, for the EES1087EP2 parameter set where $N = 1087$, $q = 2048$ and $d = 120$, the failure probability is $5.62 * 10^{-78}$, which is a bit less than 2^{-256}). We can choose the parameters (d, n, q), so that $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)} = 0$ in which that, the selected parameters should be have to satisfy following condition[2, 13] to zero this probability (the condition shown below, get from [2]),

$$d_f < \frac{\frac{q}{2} - 1}{4 * p} - \frac{1}{2} \qquad (6)$$

also, we tested a C language implementation of this condition, and approve that for $q > 2^8$, $p = 3$, d_f with the above condition and every values of n, $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ become 0 (although lower bound $q > 2^8$ not mentioned in the [2], the proof not be affected for other more exact bounds of q from valid references, since in this proof, the selected parameter set be assumed with lowest effect on the proof process). At result, we have the following rule:

$$(\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)} == 0) \rightarrow (Width(a) < q) \quad (7)$$

IV. Formalization of Ntruencrypt Scheme

What follows is a summary of the most important steps of the formalization of NTRUEncrypt Scheme in Isabelle/HOL in order to introduce suitable framework of ingredients for NTRUEncrypt algorithm, that make it useful for different functional and security analysis in future.

Formalizing of the NTRUEncrypt involved with different concepts that the researches such as, *Proving Real-Valued Inequalities by Computation in Isabelle/HOL*[11], *Proofs of properties of finite-dimensional vector spaces using Isabelle/HOL*[15], *Defining Recursive Functions in Isabelle/HOL*[18], and *Executable matrix operations on matrices of arbitrary dimensions*[19], help us to handle them in the design. Also Isabelle libraries introduced useful $Theory$ files that we can *import* them in our theory (such as list, vector list, ring and...)[7, 17], but since use of them make the proof process so much harder and more complex, it is decided to use them as least as possible, in other words, most of definitions in this research were tailored to the application and thus enabled short and elegant proofs that are not easily possible with the existing libraries (that is, this thesis just specialized for NTRUEncrypt family). We implement the problem in 3 forms: *1- (just)* $apply\_script$ *lemmas*, *2- locales*[14] and *3- classes*, at the result since the verification of this encryption scheme is very heavy, so we select the well-structured Isabelle Isar proofs with the short step[10] in proving and underlying hierarchy of algebraic classes[7, 14-15] in implementing the operations and lemmas on the convolution modular lattices. Note that type classes are only useful if they are instantiated for several types, but in this research, only integers to be needed. In fact, philosophy of using the type classes in this research is classification of formal primitives to introduce more readability in proof process!

The $theorem\ NTRUEncrypt_{Verification}$ is the main theorem of our thesis. Proving of this theorem consists of 8 main steps (main goals), that every main steps may have some sub steps (subgoals). For more readability of proving steps, the assumptions in $theorem\ NTRUEncrypt_{Verification}$ come with "$\_assm$" in their names and the lemmas used in the theorem proving come with "$\_rule$" in their names.

As will be shown in the codes, the "priority order" set by a number that specify the priority, such as $(infixl\ +70)$, so that big number cause the high priority. Following sample equality shows the priority order of operations in this research (the operator »i that return the i-th integer element of a vector, which is not discussed in this paper):

$$-\varphi \otimes p \cdot h^{-1} \oplus f»5\ \cdot\ m\ mod\ q\ \equiv$$
$$(-(\varphi \otimes (p \cdot (h^{-1}))) \oplus ((f»5) \cdot m))\ mod2\ (q) \quad (8)$$

Note that we omit the unnecessary declarations, definitions, functions and lemmas in this text to make the thesis more convenient to read.

A. Basic Primitives

Basic concepts in this formalization is the implementation of vectors with same length, addition and multiplication operation on them. The easiest way to achieve this goal is that, preconditions about the length of vectors being omnipresent with the use of unspecified constant definition. So the first basic primitive is declaration of n as a constant, that show length of vectors,
$\boldsymbol{consts}\ n :: "nat"$

Since, fixed length condition of vectors is not necessary in addition operation to satisfy correctness of related rules and assumptions in this $Theory$, we implemented it without length constraint of operands,

```
fun $plus_{int\_list}$
:: "$int\ list \Rightarrow int\ list \Rightarrow int\ list (infixl\ +70)$" $where$
    "$[] + [] = []$" |
    "$(x\#xs) + [] = x\ \#\ (xs + [])$" |
    "$[] + (y\#ys) = y\ \#\ ([] + ys)$" |
    "$(x\#xs) + (y\#ys) = (x + y)\ \#\ (xs + ys)$"
```

Multiplication operation on lattice vectors, against the addition operation, is a complex recursive function that implement matrix vector multiplication (described in section 3.1), and the correctness of it rely on fixed length of vector operands, so we should apply this length constraint on the vectors for multiplication (and other multiplication related operations and rules) with the definition of following function,

```
fun $n_{dim\_vector}$ :: "$nat \Rightarrow int\ list \Rightarrow int\ list$" $where$
    "$n_{dim\_vector}\ 0\ [] = []$"|
    "$n_{dim\_vector}\ 0\ xs = []$"|
    "$n_{dim\_vector}\ (Suc\ i)\ [] = (0\ \#\ (n_{dim\_vector}\ i\ []))$"|
    "$n_{dim\_vector}\ (Suc\ i)\ (x\ \#\ xs) =$
      $(x\ \#\ n_{dim\_vector}\ i\ xs)$"
```

This function assume that we reduced a polynomial in the modulo of $X^n - 1$, so we just have polynomials of degree $n - 1$ and less (vectors with the length of $\leq n$) and because all the operations in this $Theory$ have not vector outputs of length $> n$, so we should only expand the length vectors with the zero entries in the most significant entries up to n entries. Implementation of the function that compute a polynomial modulo of $X^n - 1$, have least usefulness and on the other hand heavily increased the proof processes. Since we implement the lattice integer vectors in a hierarchy of classes, so we apply the length constraint on this lattice vectors with the definition of base class $vector$ and inherit from it,

```
class $vector = fixes\ vector$
    :: "'a list ⇒'a list" ($\mathcal{V}$ _[50])
    $assumes\ length_{rule}[simp]$: "$length(\mathcal{V}\ A) = n$"
instantiation $int :: vector$
$begin$
    $definition\ vector_{int}$:: "int list ⇒ int list" $where$
    $\mathcal{V}_{def}$ : "$\mathcal{V}\ A\ \equiv\ n_{dim\_vector}\ n\ A$";
    $<instance\ proof\ ...>$
$end$
```

Since Isabelle2009-1, the Isabelle/HOL library already provides a type of lists of length n', namely "vec" in HOL/Multivariate_Analysis/Finite_Cartesian_Product with instances for all the required group classes but, as mentioned at the beginning of this section, this thesis just specialized for NTRUEncrypt family to introduce short and elegant proofs (that are not easily possible with the existing libraries). Even if we decided to use the Isabelle/HOL library, the ring classes of this research would have to be instantiated manually, because the multiplication operation is specific to NTRUEncrypt. The rec_{mult} function implement Matrix vector multiplication with the use of $inner_{mult\_vec}$ (note that inputs of these two recursive function is vectors of length n),

```
fun $inner_{mult\_vec}$ :: "$int\ vec \Rightarrow int\ vec \Rightarrow int$" $where$
```

$"inner_{mult\_vec}\ [][] = 0"|$
$"inner_{mult\_vec}\ (b\#B)\ (c\#C) = (b * c) + (inner_{mult\_vec}\ B\ C)"$
$\boldsymbol{fun}\ rec_{mult} :: "int\ vec \Rightarrow int\ vec \Rightarrow nat \Rightarrow int\ vec"\ where$
$"rec_{mult}\ A\ B\ 0\ =\ []"|$
$"rec_{mult}\ A\ B\ (Suc\ i) = (inner_{mult\_vec}\ \ A\ B)\ \# (rec_{mult}\ (rotate\ (n-1))\ A)\ B\ i)"$

Since the underlying constructs for operation on convolution modular lattices implement with the hierarchy of algebraic classes, we should define inverse elements of ⊗ and ⊕ with functions of inv_{plus_vec} and inv_{mult_vec} (we define operation of inv_{mult_vec} in the form of some propositions):

$\boldsymbol{fun}\ neg_{int\_vec} :: "int\ vec\ \Rightarrow\ int\ vec"\ where$
$"neg_{int\_vec}\ []\ =\ []"\ |$
$"neg_{int\_vec}\ (x\#xs)\ =\ (-x)\ \#\ neg_{int\_vec}\ \ xs"$
$\boldsymbol{consts}\ inv_{plus\_vec} :: "'a\ vec \Rightarrow 'a\ vec"\ ("-\_"\ [81]\ 80)$
$\boldsymbol{defs}\ inv_{plus\_vec\_def}: "inv_{plus\_vec}\ xs\ \equiv\ neg_{int\_vec}\ xs"$
$\boldsymbol{consts}\ inv_{mult\_vec} :: "'a\ vec \Rightarrow 'a\ vec"\ ("\_^{-1}"\ [111]\ 110)$

We can apply a coefficient for a vector in which that multiply the value of the coefficient with every entries of that vector,

$\boldsymbol{definition}\ Scale :: "int \Rightarrow int\ vec \Rightarrow int\ vec"\ (infix\ "\cdot"\ 100)\ where$
$"x\ \cdot\ xs\ \equiv\ map\ (op\ *\ x)\ xs"$

In this text, we show the operator modulo of an Integer on an integer and modulo of a vector on an integer with the same notation mod (that defined as following function),

$\boldsymbol{fun}\ mod_{list} :: "int\ vec \Rightarrow int \Rightarrow int\ vec"\ (infixl\ "mod"\ 60)\ where$
$"[b]\ mod\ c\ =\ [b\ mod\ c]"|$
$"(b\#B)\ mod\ c\ =\ ((b\ mod\ c)\#(B\ mod\ c))"$

B. Types Declaration/Definition

In this subsection we just introduced the Declarations and Definitions that used in formalization and verification:

- The annotation vec is just to imply operation on vectors,
  $\boldsymbol{type\_synonym}\ 'a\ vec\ =\ "'a\ list"$

- The four variables n, d, p and q used in 3 form nat, int and rat so we implement 2 function $nat2int$ and $nat2rat$ to handle type conversions (in this paper each 3 forms of these variables used in same way),
  $\boldsymbol{consts}\ \ d :: "nat";$
  $\boldsymbol{consts}\ \ q :: "nat";$
  $\boldsymbol{consts}\ \ p :: "nat";$

- $ZERO$ vector is the unit element of class $monoidl_{plus}$ (discussed in subsection classes hierarchy),
  $\boldsymbol{consts}\ ZERO :: "'a\ vec"\ ("\mathbf{0}")$
  $\boldsymbol{defs}\ \ ZERO_{def}: "ZERO\ \equiv\ \mathcal{V}\ []"$

- ONE vector is the unit element of class $monoidl_{mult}$ (discussed in subsection classes hierarchy),
  $\boldsymbol{consts}\ ONE :: "'a\ vec"\ ("\mathbf{1}")$
  $\boldsymbol{defs}\ \ ONE_{def}: "ONE\ \equiv\ \mathcal{V}\ [1]"$

- Following vectors with the same notation discussed in section 3,
  $\boldsymbol{consts}\ f :: "int\ vec"$
  $\boldsymbol{consts}\ g :: "int\ vec"$
  $\boldsymbol{consts}\ h :: "int\ vec"$
  $\boldsymbol{consts}\ e :: "int\ vec"$
  $\boldsymbol{consts}\ a :: "int\ vec"$
  $\boldsymbol{consts}\ m :: "int\ vec"$
  $\boldsymbol{consts}\ r :: "int\ vec"$
  $\boldsymbol{consts}\ decrypted_m :: "int\ vec"$
  $\boldsymbol{consts}\ M :: "int\ vec"$
  $\boldsymbol{consts}\ R :: "int\ vec"$
  $\boldsymbol{definition}\ Fq :: "int\ vec"\ where$
  $"Fq = f^{-1} mod\ q"$
  $\boldsymbol{definition}\ Fp :: "int\ vec"\ where$
  $"Fp = f^{-1} mod\ p"$

- Following boolean variables specify certain conditions in the decryption algorithm and with the same notation discussed in section 3,
  $\boldsymbol{consts}\ Centring_{Start} :: "bool"$
  $\boldsymbol{consts}\ gap_{failure}\ \ \ :: "bool"$
  $\boldsymbol{consts}\ wrap_{failure}\ \ \ :: "bool"$

- Following functions with the same notation discussed in section 3,
  $\boldsymbol{consts}\ \mathcal{E} :: "int\ vec \times int\ vec \Rightarrow int\ vec \times int\ vec"$
  $\boldsymbol{consts}\ \mathcal{E}_{Inv} :: "int\ vec \times int\ vec \Rightarrow int\ vec \times int\ vec"$
  $\boldsymbol{consts}\ \mathcal{R} :: "int\ vec\ \Rightarrow\ int\ vec"$

- $small_{coef}$ is set of possible coefficients for f, g, m and r,
  $\boldsymbol{definition}\ small_{coef} :: "int\ set"\ where\ "small_{coef} = \{-1, 0, 1\}"$

This research used lots of unspecified constants ($\boldsymbol{consts}$) to simplify the proof steps. Consequently, inside HOL, the theorem holds only for those unspecified values (Only on the meta-level, when one considers the set-theoretic semantics of HOL, one regains generality by considering different models for the constants).

C. Classes Hierarchy

As mentioned, because of complexity of operations in the convolution modular lattice vectors, we implement these vectors in a classes hierarchy design, as shown in s.1:

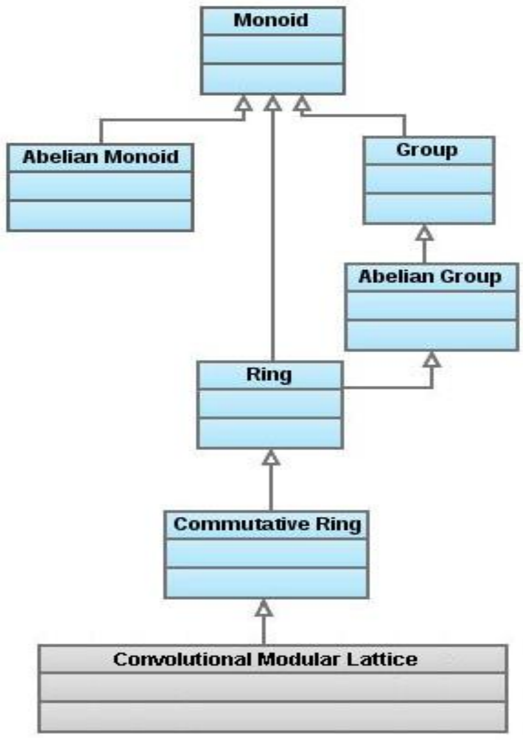

Fig.1. Diagram of class inheritance for Convolution Modular Lattice

The algebraic type class hierarchy introduced in this subsection, is nearly similar to the algebraic class hierarchies distributed with Isabelle/HOL but the main difference is that the class operations here, refer directly to lists of length n, where n is an unspecified constant. This subsection divide in to 14 part that in each part we define a new class, instantiate it on int type and also, sometimes related lemmas mentioned (zero integer in this text showed by 0 and zero vector of length n showed by $\mathbf{0}$, also unit integer showed by 1 and unit vector of the introduced multiplication with length n showed by $\mathbf{1}$).

part1) $vector$: This class is the base class and other classes inherit the vector length constraints from it (as shown before).

part2) $p\,lus$: This class inherit from $vector$, but because the operation $\oplus$ it don't depend on length of vector, so don't apply its length constraint (with $\mathcal{V}$ prefix),

$\boldsymbol{class}\; plus\; =\; vector +$
$\quad fixes\; plus$
$\quad :: \text{"}'a\; vec \Rightarrow 'a\; vec \Rightarrow 'a\; vec\text{"}(infixl\; \text{"}\oplus\text{"}\; 70)$
$\boldsymbol{instantiation}\; int :: plus\; begin$
$\quad definition\;\; plus_{int}$
$\quad :: \text{"}int\; vec\;\; \Rightarrow\;\; int\; vec \Rightarrow int\; vec\text{"}$
$\quad where\; \text{"}A \oplus B \equiv A + B\text{"}\quad instance\; ..\, end$

part3) $semigroup_{plus}$: This class inherit from $plus$, and apply the rule of right and left associative on the $\oplus$ operation,

$\boldsymbol{class}\; semigroup_{plus}\; =\; plus\; +$
$\quad assumes\; semigroup_{assoc\_right\_plus}\!: \text{"}(x \oplus y) \oplus z\; =\; x \oplus (y \oplus z)\text{"}$
$\boldsymbol{lemma}\; (in\; semigroup_{plus})\; semigroup_{assoc\_left\_plus}\!:$
$\quad shows\; \text{"}(x) \oplus (y \oplus z)\; =\; (x \oplus y) \oplus z\text{"}$
$\boldsymbol{instantiation}\; int :: semigroup_{plus}\; begin$
$\quad < instance\; proof\; \ldots > end$

part4) $monoidl_{plus}$: This class inherit from $semigroup_{plus}$, and apply the rule of left unit element on the $\oplus$ operation (note that in this class, rule of $monoidl_{Zero\_plus}$ need to apply length constraint, so we use from this constraint by $\mathcal{V}$ prefix),

$\boldsymbol{class}\; monoidl_{plus} = semigroup_{plus} +$
$\quad assumes\; monoidl_{Zero\_plus}\!: \text{"}\mathbf{0} \oplus (\mathcal{V}\; A)\; =\; (\mathcal{V}\; A)\text{"}$
$\boldsymbol{instantiation}\; int :: monoidl_{plus}\; begin$
$\quad < instance\; proof\; \ldots >\;\; end$

Part5) $monoid_{plus}$: This class inherit from $monoidl_{plus}$, and apply the rule of right unit element on the $\oplus$ operation (note that in this class, rule of $monoidr_{Zero\_plus}$ need to have length constraint, so we use from this constraint by $\mathcal{V}$ prefix),

$\boldsymbol{class}\; monoid_{plus} = monoidl_{plus} +$
$\quad assumes\; monoidr_{Zero\_plus}\!: \text{"}(\mathcal{V}\; A) \oplus \mathbf{0} = (\mathcal{V}\; A)\text{"}$
$\boldsymbol{instantiation}\; int :: monoid_{plus}\; begin$
$\quad < instance\; proof\; \ldots > end$

part6) $group_{plus}$: This class inherit from $monoid_{plus}$, and apply the rule of inverse element (and following this rule, $group_{plus}$ satisfy $group_{left\_cancel\_plus}$ rule) on the $\oplus$ operation,

$\boldsymbol{class}\; group_{plus} = monoid_{plus} +$
$\quad assumes\; group_{Inverse\_plus}\!:$
$\quad \text{"}(-(\mathcal{V}\; A)) \oplus (\mathcal{V}\; A)\; =\; \mathbf{0}\text{"}$
$\boldsymbol{lemma}\; (in\; group_{plus}) inverse_{equality}\!:$
$\quad \text{"} - (\mathcal{V}\; A) = \mathcal{V}(-A)\text{"}$
$\boldsymbol{lemma}\; (in\; group_{plus}) group_{left\_cancel\_plus}\!:$
$\quad \text{"}(\mathcal{V}\; A) \oplus (\mathcal{V}\; B) = (\mathcal{V}\; A) \oplus (\mathcal{V}\; C) \leftrightarrow (\mathcal{V}\; B) = (\mathcal{V}\; C)\text{"}$
$\boldsymbol{instantiation}\; int :: group_{plus}\; begin$
$\quad < instance\; proof\; \ldots > end$

part7) $abelian_{group\_plus}$: This class inherit from $group_{plus}$ and apply the rule of commutative on the $\oplus$ operation,

$\boldsymbol{class}\; abelian_{group\_plus}\; =\; group_{plus}\; +$
$\quad assumes$
$\quad abelian_{group\_left\_cancel\_plus}\!: \text{"}A \oplus B\; =\; B \oplus A\text{"}$
$\boldsymbol{instantiation}\; int :: abelian_{group\_plus}\; begin$
$\quad < instance\; proof\; \ldots >\;\; end$

part8) $mult$: This class inherit from $vector$, and because the operation $\otimes$ fully depend on length of vector, we apply it's length constraint (with $\mathcal{V}$ prefix),

$$
\begin{array}{l}
\boldsymbol{class}\ mult\ =\ vector\ +\\
\quad fixes\ mult\\
\quad ::\ "'a\ vec \Rightarrow 'a\ vec \Rightarrow 'a\ vec"\ (infixl\ "\otimes"\ 90)\\
\boldsymbol{instantiation}\ int :: mult\ begin\\
\quad definition\ \ mult_{int}\\
\quad :: "int\ vec \Rightarrow int\ vec \Rightarrow int\ vec"\ where\\
\qquad "A \otimes B \equiv\\
\qquad rec_{mult}(rotate(n-1)(rev\ (\mathcal{V}\ A)))\ (\mathcal{V}\ B)\ n"\\
\quad instance\ ..\ end
\end{array}
$$

part9) $semigroup_{mult}$: This class inherit from $mult$, and apply the rule of right and left associative on the $\otimes$ operation,

$$
\begin{array}{l}
\boldsymbol{class}\ semigroup_{mult} = mult\ +\\
\quad assumes\ \ semigroup_{assoc\_right\_mult}:\\
\quad "(A \otimes B) \otimes C\ =\ A \otimes (B \otimes C)"\\
\boldsymbol{lemma}\ (in\ semigroup_{mult})\ semigroup_{assoc\_left\_mult}:\\
\quad "A \otimes (B \otimes C)\ =\ (A \otimes B) \otimes C"\\
\boldsymbol{instantiation}\ int :: semigroup_{mult}\ begin\\
\quad < instance\ proof\ ... >\ \ end
\end{array}
$$

part10) $monoidl_{mult}$: This class inherit from $semigroup_{mult}$, and apply the rule of left unit element on the $\otimes$ operation,

$$
\begin{array}{l}
\boldsymbol{class}\ monoidl_{mult} = semigroup_{mult}\ +\\
\quad assumes\ monoidl_{Zero\_mult}: "\mathbf{1} \otimes A\ =\ A"\\
\boldsymbol{instantiation}\ int :: monoidl_{mult}\ begin\\
\quad < instance\ proof\ ... >\ \ end
\end{array}
$$

part11) $monoid_{mult}$: This class inherit from $monoidl_{mult}$, and apply the rule of right unit element on the $\otimes$ operation,

$$
\begin{array}{l}
\boldsymbol{class}\ monoid_{mult} = monoidl_{mult}\ +\\
\quad assumes\ monoid_{Zero\_mult}: "A \otimes \mathbf{1}\ =\ A"\\
\boldsymbol{instantiation}\ int :: monoid_{mult}\ begin\\
\quad < instance\ proof\ ... >\ end
\end{array}
$$

part12) $ring$: This class inherit from $monoid_{mult}$ and $abelian_{group\_plus}$(as shown in Fig. 1) and apply the rule of left distributive and right distributive of $\otimes$ operation on the $\oplus$ operation,

$$
\begin{array}{l}
\boldsymbol{class}\ ring\ =\ monoid_{mult} + abelian_{group\_plus}\ +\\
\quad assumes\ ring_{distributivity\_left}\ :\\
\quad "A \otimes (B \oplus C)\ =\ A \otimes B \oplus A \otimes C"\\
\boldsymbol{lemma}\ (in\ ring) ring_{distributivity\_right}\\
\quad : "(A \oplus B) \otimes C\ =\ A \otimes C \oplus B \otimes C"\\
\boldsymbol{instantiation}\ int :: ring\ begin\\
\quad < instance\ proof\ ... >\ end
\end{array}
$$

part13) $commutative_{ring}$: This class inherit from $ring$ (as shown in Fig. 1) and apply the rule of commutative on the $\otimes$ operation,

$$
\begin{array}{l}
\boldsymbol{class}\ commutative_{ring} = ring\ +\\
\quad assumes\ commutative_{ring\_rule} : "A \otimes B\ =\ B \otimes A"\\
\boldsymbol{instantiation}\ int :: commutative_{ring}\ begin\\
\quad < instance\ proof\ ... >\ end
\end{array}
$$

part14) $conv_{mod\_lat}$ (Convolution modular Lattice): This class just inherit from $commutative_{ring}$(as shown in Fig. 1),

$$
\begin{array}{l}
\boldsymbol{class}\ conv_{mod\_lat} = commutative_{ring}\\
\boldsymbol{instantiation}\ int :: conv_{mod\_lat}\ begin\\
\quad instance..end
\end{array}
$$

D. Defined functions in $conv_{mod\_lat}$ class

Encryption/Decryption related functions specifically operate on the $conv_{mod\_lat}$ class that can be defined in the following way:

- Encryption method defined as function $\mathcal{ENC}_{\mathcal{M}}$,

$$
\begin{array}{l}
\boldsymbol{definition}\ (in\ conv_{mod\_lat})\mathcal{ENC}_{\mathcal{M}} ::\\
"int \Rightarrow int\ vec \Rightarrow int\ vec \Rightarrow int\ vec \Rightarrow int \Rightarrow\\
int\ vec"\ where\\
"\mathcal{ENC}_{\mathcal{M}}\ p'\ r'\ h'\ m'\ q' \equiv\\
(p' \cdot r' \otimes h' \oplus m')\ mod\ q'"
\end{array}
$$

- Public key compute as function $\mathcal{H}$,

$$
\begin{array}{l}
\boldsymbol{definition}\ (in\ conv_{mod\_lat})\ \mathcal{H} ::\\
"int\ vec\ \Rightarrow\ int\ vec\ \Rightarrow\ int\ \Rightarrow\ int\ vec"\ where\\
"\mathcal{H}\ Fq'\ g'\ q'\ =\ (Fq' \otimes g')\ mod\ q'"
\end{array}
$$

- We can compute a (from step 1 of decryption process) as the following function:

$$
\begin{array}{l}
\boldsymbol{definition}\ (in\ conv_{mod\_lat})\ \mathcal{A}\\
:: "int\ vec \Rightarrow int\ vec \Rightarrow int\ vec"\ where\\
"\mathcal{A}\ e'\ f' \equiv\ e' \otimes f'\ mod\ q'"
\end{array}
$$

- Decryption method defined as function $\mathcal{DEC}_{\mathcal{M}}$,

$$
\begin{array}{l}
\boldsymbol{definition}\ (in\ conv_{mod_{lat}})\mathcal{DEC}_{\mathcal{M}} ::\\
"int\ vec\ \Rightarrow\ int\ vec\ \Rightarrow\ int\ \Rightarrow\ int\ vec"\ where\\
"\mathcal{DEC}_{\mathcal{M}}\ Fp'\ a'\ p'\ \equiv\ (Fp' \otimes a')\ mod\ p'"
\end{array}
$$

- Width method defined as function $Width$,

$$
\begin{array}{l}
\boldsymbol{definition} (in\ conv_{mod\_lat}) Width\\
:: "int\ vec \Rightarrow int"\ where\\
"Width\ X\ =\ ((Max\ (set\ X)) - (Min\ (set\ X)))"
\end{array}
$$

V. Verification of Ntru Encryption Scheme

What follows is a summary of the most important steps (including necessary lemmas and assumptions) of the formal proof in Isabelle/HOL in order to outline the general course. In the next subsections we described necessary lemmas and assumptions that used in proof of theory $NTRUEncrypt_{Verification}$, after we discussed the implementation of Encryption/Decryption scheme (implemented as assumption steps in $NTRUEncrypt_{Verification}$) and in last subsection, we discussed briefly the proof steps of $NTRUEncrypt_{Verification}$ (note that all the lemmas and assumptions in this section defined in the context of class $conv_{mod\_lat}$, so we omit the command $in\ conv_{mod\_lat}$ from these lemmas).

A. Lemma about the $Scale$ function

Function $Scale$ introduced two important lemmas which defined as,

$\boldsymbol{lemma}\ vector_{scaling\_rule1}$:
"$A \otimes (S \cdot B) = (S \cdot A) \otimes B$"
$\boldsymbol{lemma}\ vector_{scaling\_rule2}$: "$X \cdot A\ mod\ X = 0$"

B. Lemma about the $length$ constraint

As mentioned, the vectors in convolution modular lattice should be in the fixed length n and operator $\mathcal{V}$ apply this length constraint on the our vectors,

$\boldsymbol{lemma}\ length_{rule\_lemma1}$:
"$[\![A = B]\!] \Longrightarrow (length\ A = length\ B)$"
$\boldsymbol{lemma}\ length_{rule\_lemma2}$:
"$[\![A = B]\!] \Longrightarrow (\mathcal{V}\ A = \mathcal{V}\ B)$"
$\boldsymbol{lemma}\ length_{rule\_lemma3}$:
"$[\![(A::int\ vec) = \mathcal{V}\ A]\!] \Longrightarrow (length\ A = n)$"
$\boldsymbol{lemma}\ length_{rule\_lemma4}$: "$length(\mathcal{V}\ A) = n$"

and for each convolutional modular lattice vectors (f, g, h, e, a, m, r, $decrypted\_m$, m', Fq, Fp, $centered\_m$, $\mathbf{0}$, $\mathbf{1}$, M and R) we define this sample lemma (substitute X with the vector name, such as f):

$\boldsymbol{lemma}\ length_{rule\_lemmaX}$: $assumes\ assm$: "$X = \mathcal{V}\ X$"
$shows$ "$length\ X = n$"

C. Lemma about multiplication inverse element

A convolution modular lattice is a ring and each element of this construct may have or have not inverse element so we defined the multiplication inverse operation as the following lemma:

$\boldsymbol{lemma}\ ring_{mult\_inverse\_rule}$:
"$[\![\exists\ X. (X = A^{-1}\ mod\ B)]\!] \Longrightarrow$
$(X \otimes A\ mod\ B = \mathbf{1} \wedge (A \otimes X)\ mod\ B = \mathbf{1})$"

D. Lemma about the mod operation

Since operation on the algebraic constructions that defined by mod operation, introduced longer proof step to handle and work on the propositions, so we defined a set of rules to shortening these proof steps (by guiding the automatic proof tools):

$\boldsymbol{lemma}\ mod_{rule1}$:
"$(A\ mod\ X)\ mod\ X = A\ mod\ X$"
$\boldsymbol{lemma}\ mod_{rule2}$:
"$(A\ mod\ X) \otimes B\ mod\ X = A \otimes B\ mod\ X$"
$\boldsymbol{lemma}\ mod_{rule3}$:
"$[\![A = B]\!] \Longrightarrow (A\ mod\ X = B\ mod\ X)$"
$\boldsymbol{lemma}\ mod_{rule4}$:
"$A \otimes B\ mod\ X = ((A\ mod\ X) \otimes B)\ mod\ X$"
$\boldsymbol{lemma}\ mod_{rule5}$:
"$(A \oplus B)\ mod\ X = ((A\ mod\ X) \oplus B)\ mod\ X$"
$\boldsymbol{lemma}\ mod_{rule6}$:
"$[\![(A\ mod\ X) = B]\!] \Longrightarrow (A\ mod\ X = B\ mod\ X)$"
$\boldsymbol{lemma}\ mod_{rule7}$:
"$(A\ mod\ X) \otimes (B\ mod\ X)\ mod\ X = A \otimes B\ mod\ X$"

E. Assumptions about n, p, q

As be mentioned, we have 3 important integer parameter that should apply some assumptions on them,

$n\_def_{assm}[intro]$: "$n > 0$"
$modul_{assm1}$: "$gcd(p, q) = 1$"
$modul_{assm2}$: "$q > p$"

F. Assumption about $Width$ of m, r, f, g

For simplicity, according to definition of NTRUEncrypt in [2], and with the assumption of $p = 3$ ($Skip_{assm3}$), if we assume the following assumptions:

$mod_{assm\_modp\_m}$: "$m = m\ mod\ p$"
$mod_{assm\_modp\_r}$: "$r = r\ mod\ p$"
$mod_{assm\_modp\_f}$: "$f = f\ mod\ p$"
$mod_{assm\_modp\_g}$: "$g = g\ mod\ p$"

the assumptions about $Width$ of vectors m, r, f, g (that equal to 2), and coefficients of these vectors (that belong to $small_{coef}$), can be eliminated (that is the following assumptions be eliminated):

$coef_{assm\_m}$: "$set\ m = small_{coef}$"
$coef_{assm\_r}$: "$set\ r = small_{coef}$"
$coef_{assm\_f}$: "$set\ f = small_{coef}$"
$coef_{assm\_g}$: "$set\ g = small_{coef}$"
$width_{assm\_m}$: "$Width(m) = 2$"
$width_{assm\_r}$: "$Width(r) = 2$"
$width_{assm\_f}$: "$Width(f) = 2$"
$width_{assm\_g}$: "$Width(g) = 2$"

G. Assumption about Encoding Scheme

Since the encoding and decoding scheme just be declared, so we use the following assumptions to define their operations,

$Encoding_{Scheme\_assm1}$:
"$[\![\mathcal{E}(R', M') = (r', m')]\!] \Longrightarrow (\mathcal{E}_{Inv}\ (r', m') = (R', M'))$"
$Encoding_{Scheme\_assm2}$:
"$[\![\mathcal{E}(R', M') = (r', m')]\!] \Longrightarrow (\mathcal{E}_{Inv}\ ([i], m') = (R', M'))$"

H. Assumption about Probability of Decryption Failure

We implement the $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ in Isabelle/HOL, but since the proving process with the operation on rat in this complex function ($\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$) is beyond our thesis[7, 11] and also if we bound the failure probability, it may be required to switch to probabilistic reasoning (in that direction, the theory HOL/Probability/Probability_Mass_Function might be a good starting point). So we assume that parameters d, n and q select from a specific parameter set that zeroed the $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ value (discussed in subsection 3.4), and consequently add the following assumption (note that if $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ be zeroed, we have not $gap_{failure}$):

$Skip_{assm1}$: "$q > 2^8$"
$Skip_{assm2}$: "$d < \frac{\frac{q}{2} - 1}{4 * p} - \frac{1}{2}$"
$Skip_{assm3}$: "$p = 3$"
$Parameter\_Set_{assm}$:
"$[\![(q > 2^8);\ (d < \frac{\frac{q}{2}-1}{4*p} - \frac{1}{2})\ ;(p = 3)]\!] \Longrightarrow \mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)} = 0$"

Assumptions $Skip_{assm1}$, $Skip_{assm2}$, $Skip_{assm3}$ and $Parameter\_Set_{assm}$ correspond with phases of passing the gap failure, that is, $NTRUEncrypt_{Decryption\_phase2}$, $NTRUEncrypt_{Decryption\_phase3}$ and $NTRUEncrypt_{Decryption\_phase4}$, so that proof system pass out this three steps of decryption phases automatically.

I. Assumption about the centering method

If we have not $gap_{failure}$ (since we assume that the parameters selected based on $Parameter\_Set_{assm}$ cause the $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ be zeroed), the coefficients of r, g, m, f are small, so the coefficients of $p.r \otimes g \oplus m \otimes f$ will lie in an interval of length less than q. We assume always centring method chooses the appropriate interval, so the polynomial a equals $p.r \otimes g \oplus m \otimes f$ exactly, and not merely modulo q, so the $wrap_{failure}$ not be happened. The following assumption have important role in the proof process:

$centring_{method\_assm}$:
"$[\![gap_{failure} = False; wrap_{failure} = False]\!] \Longrightarrow (\mathcal{R}\ a\ =\ (p \cdot r \otimes h \oplus m\ mod\ q) \otimes f)$"

Assumption $centring_{method\_assm}$ correspond with $NTRUEncrypt_{Decryption\_phase5}$, so that proof system pass out this step of decryption phases automatically. Although, transition from Z/q to Z is again assumed instead of proven, but we should note that proving the success of centering method has so complex states!

J. Encryption Process

We implement the steps of encryption process as assumptions of $NTRUEncrypt_{Verification}$ in the following 3 phases (M is original input message):

$NTRUEncrypt_{Encryption\_phase1}$: "$h\ =\ \mathcal{H}\ Fq\ g\ q$"
$NTRUEncrypt_{Encryption\_phase2}$: "$\mathcal{E}\ (R\ , M)\ =\ (r\ , m)$"
$NTRUEncrypt_{Encryption\_phase3}$: "$e = \mathcal{ENC}_{\mathcal{M}}\ p\ r\ h\ m\ q$"

K. Decryption Process

We implement the steps of decryption process as assumptions of $NTRUEncrypt_{Verification}$ in the following 7 phases:

$NTRUEncrypt_{Decryption\_phase1}$: "$a\ =\ (\mathcal{A}\ e\ f)$"
$NTRUEncrypt_{Decryption\_phase2}$:
"$[\![\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)} = 0]\!] \Longrightarrow (Width(a) < q)$"
$NTRUEncrypt_{Decryption\_phase3}$:
"$[\![Width(a) < q]\!] \Longrightarrow (gap_{failure} = False)$"
$NTRUEncrypt_{Decryption\_phase4}$:
"$[\![gap_{failure} = False]\!] \Longrightarrow (Centring_{Start} = True)$"
$NTRUEncrypt_{Decryption\_phase5}$:
"$[\![Centring_{Start} = True]\!] \Longrightarrow (wrap_{failure} = False)$"
$NTRUEncrypt_{Decryption\_phase6}$:
"$[\![wrap_{failure} = False]\!] \Longrightarrow (decrypted_m = \mathcal{DEC}_{\mathcal{M}}\ Fp\ (\mathcal{R}\ a)\ p)$"
$NTRUEncrypt_{Decryption\_phase7}$:
"$\mathcal{E}_{Inv}([\ ], decrypted_m) = (R', M')$"

L. Roadmap of formal verification for NTRUEncrypt

Theorem $NTRUEncrypt_{Verification}$ defined in the context of $conv_{mod\_lat}$. As be mentioned, we used a forward proof steps by Isabelle/Isar structure. We outline proof steps of $NTRUEncrypt_{Verification}$ in the following way (note that the following proof road map is not Isabelle/HOL commands and just show outline the proof steps):

$\overset{use}{\Longrightarrow} [\![Skip_{assm1} \wedge Skip_{assm2} \wedge Skip_{assm3} \wedge Parameter_{Set\_assm}]\!]$
$\longrightarrow (NTRUEncrypt_{Decryption\_phase1})$
$\longrightarrow (Width(a) < q)$

$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase2}\right)$
$\longrightarrow \left(gap_{failure} = False\right)$
$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase3}\right)$
$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase4}\right)$
$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase5}\right)$
$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase6}\right)$
$\overset{use}{\Longrightarrow} [\![vector_{scaling\_rule1} \wedge$
$vector_{scaling\_rule2} \wedge length_{rule\_lemma1} \wedge$
$length_{rule\_lemma2} \wedge length_{rule\_lemma3} \wedge$
$length_{rule\_lemma4} \wedge length_{rule\_lemmaX} \wedge$
$ring_{mult\_inverse\_rule} \wedge mod_{rule1} \wedge mod_{rule2} \wedge$
$mod_{rule3} \wedge mod_{rule4} \wedge mod_{rule5} \wedge mod_{rule6} \wedge$
$mod_{rule7} \wedge mod_{assm\_modp\_m} \wedge \ od_{assm\_modp\_r} \wedge$
$mod_{assm\_modp\_f} \wedge mod_{assm\_modp\_g} \wedge$
$mod_{assm\_modq\_m} \wedge mod_{assm\_modq\_r} \wedge$
$mod_{assm\_modq\_f} \wedge mod_{assm\_modq\_g} \wedge$
$centring_{method\_assm} \wedge commutative_{ring\_rule} \wedge$
$ring_{distributivity\_right} \wedge ring_{distributivity\_left} \wedge$
$monoid_{Zero\_mult} \wedge monoidl_{Zero\_mult} \wedge$
$semigroup_{assoc\_left\_mult} \wedge$
$semigroup_{assoc\_right\_mult} \wedge$
$abelian_{group\_left\_cancel\_plus} \wedge inverse_{equality} \wedge$
$group_{left\_cancel\_plus} \wedge monoidr_{Zero\_plus} \wedge$
$monoidl_{Zero\_plus} \wedge semigroup_{assoc\_left\_plus} \wedge$
$semigroup_{assoc\_right\_plus} \wedge length_{rule}]\!]$
$\longrightarrow \left(NTRUEncrypt_{Decryption\_phase7}\right)$
$\overset{use}{\Longrightarrow} [\![Encoding_{Scheme\_assm1} \wedge Encoding_{Scheme\_assm2}]\!]$
$\longrightarrow (M = M')$
$< \ NTRUEncrypt_{Verification}\ be\ proved >$

As seen in the roadmap of formal verification for NTRUEncrypt, it is not clear what has actually been verified. since theorem $NTRUEncrypt_{Verification}$ assume most of the interesting steps of the scheme being correct (by assumptions introduced). In fact, this research just fully validate the control flow of proposition $\mathcal{DEC}_{\mathcal{M}}\ Fp\ (\mathcal{R}\ a)\ p\ ==\ m$ as following way (and other steps validated by some assumptions):

$$\left(\overbrace{\mathcal{DEC}_{\mathcal{M}}\ Fp \left(\overbrace{\mathcal{R}\left(\overbrace{\mathcal{A}\left(\overbrace{\mathcal{ENC}_{\mathcal{M}}\ p\ r\ \overbrace{(\mathcal{H}\ Fq\ g\ q)}^{h}\ m\ q}^{e}\right) f}^{a}\right)}^{\mathcal{R}\ a}\right) p}^{decrypted_m}\right) \overset{?}{==} m$$

so that m satisfy condition $\mathcal{E}_{Inv}([\ ], m) == (R, M)$ with the original input message M. Verifying the goal of $decrypted_m == m$ includes the bellow subgoals which nearly corresponds with the $NTRUEncrypt_{Decryption\_phase7}$ (and refered to layer 3 in the table 1 as skeleton of the proof in this research):

- $\Rightarrow$ Subgoal 1: "$(p \cdot r \otimes h \oplus m\ mod\ q) \otimes f\ mod\ q = ((p \cdot r \otimes h \otimes f\ mod\ q) \oplus (m \otimes f\ mod\ q))\ mod\ q$"
- $\Rightarrow$ Subgoal 2: "$(p \cdot\ r \otimes h) \otimes f\ mod\ q = p \cdot\ r \otimes (h \otimes f)\ mod\ q$"
- $\Rightarrow$ Subgoal 3: "$p \cdot\ r \otimes (h \otimes f)\ mod\ q = (h \otimes f\ mod\ q) \otimes p \cdot\ r\ mod\ q$"
- $\Rightarrow$ Subgoal 4: "$h \otimes f\ mod\ q = (f\ mod\ q) \otimes h\ mod\ q$"
- $\Rightarrow$ Subgoal 5: "$(f\ mod\ q) \otimes (\overbrace{(f^{-1}\ mod\ q)}^{Fq} \otimes (g\ mod\ q))\ mod\ q = ((f\ mod\ q) \otimes (f^{-1}\ mod\ q)) \otimes (g\ mod\ q)\ mod\ q$"
- $\Rightarrow$ Subgoal 6: "$(\overbrace{(f\ mod\ q) \otimes (f^{-1}\ mod\ q)}^{\mathbf{1}}) \otimes (g\ mod\ q)\ mod\ q = g\ mod\ q$"
- $\Rightarrow$ Subgoal 7: "$p \cdot\ r \otimes h \otimes f\ mod\ q = p \cdot\ r \otimes g\ mod\ q$"
- $\Rightarrow$ Subgoal 8: "$\mathcal{R}\ a = (p \cdot r \otimes h \oplus m\ mod\ q) \otimes f$"
- $\Rightarrow$ Subgoal 9: "$(p \cdot r \otimes h \otimes f\ mod\ q) \oplus (m \otimes f\ mod\ q)\ mod\ q = (p \cdot r \otimes h \otimes f\ ) \oplus (m \otimes f)$"
- $\Rightarrow$ Subgoal 10: "$Fp \otimes ((p \cdot\ r \otimes h \otimes f\ ) \oplus (m \otimes f))\ = (Fp \otimes p \cdot r \otimes h \otimes f) \oplus (Fp \otimes m \otimes f)$"
- $\Rightarrow$ Subgoal 11: "$Fp \otimes ((p \cdot r \otimes h \oplus m) \otimes f\ ) = (F \otimes p \cdot\ r \otimes h \otimes f\ ) \oplus (Fp \otimes m \otimes f)$"
- $\Rightarrow$ Subgoal 12: "$(Fp \otimes p \cdot\ r \otimes h \otimes f\ ) \oplus (Fp \otimes m \otimes f)\ = (Fp \otimes m \otimes f) \oplus (Fp \otimes p \cdot\ r \otimes h \otimes f)$"
- $\Rightarrow$ Subgoal 13: "$(Fp \otimes m \otimes f)\ mod\ p = m\ mod\ p$"
- $\Rightarrow$ Subgoal 14: "$Fp \otimes ((p \cdot r \otimes h \otimes f) \oplus (m \otimes f))\ mod\ p = (m \oplus (Fp \otimes p \cdot r \otimes h \otimes f))\ mod\ p$"
- $\Rightarrow$ Subgoal 15: "$m \oplus (Fp \otimes p \cdot r \otimes h \otimes f\ )\ mod\ p = (Fp \otimes p \cdot r \otimes h \otimes f\ mod\ p) \oplus m\ mod\ p$"
- $\Rightarrow$ Subgoal 16: "$p \cdot (r \otimes h \otimes f)\ mod\ p = \mathbf{0}$"
- $\Rightarrow$ Subgoal 17: "$Fp \otimes (p \cdot (r \otimes h \otimes f))\ mod\ p = Fp \otimes ((p \cdot (r \otimes h \otimes f))\ mod\ p)\ mod\ p$"
- $\Rightarrow$ Subgoal 18: "$Fp \otimes (p \cdot (r \otimes h \otimes f)\ mod\ p)\ mod\ p = \mathbf{0}$"
- $\Rightarrow$ Subgoal 19: "$(m \oplus (Fp \otimes p \cdot r \otimes h \otimes f))\ mod\ p =\ m$"
- $\Rightarrow$ Subgoal 20: "$Fp \otimes ((p \cdot r \otimes h \otimes f) \oplus (m \otimes f))\ mod\ p = m$"
- $\Rightarrow$ Subgoal 21: "$Fp \otimes ((p \cdot r \otimes h \oplus m\ mod\ q) \otimes f\ mod\ q)\ mod\ p = Fp \otimes ((p \cdot r \otimes h \otimes f) \oplus (m \otimes f))\ mod\ p$"
- $\Rightarrow$ Subgoal 22: "$\mathcal{DEC}_{\mathcal{M}}\ Fp\ (\mathcal{R}\ a)\ p\ =\ m$

Note that the main proof steps in the roadmap essentially not be equal to phases of decryption. As an engineering design, it is best to have a high level map to handle the main problem (formal proof of NTRUEncrypt scheme) with less details. A stack of layers in formal proving for NTRUEncrypt scheme be designed in Table 1 (including layers 1 to 4) that classify the proof steps as independent as possible in this research (this layering stack is not essentially an ideal stack for layers of formal proof of NTRUEncrypt scheme and it is possible to have better layering for this scheme):

Table 1. Stack of Layers in Formal Proof of NTRUEncrypt

$\boxed{4}$	Encoding/Decoding layer is last layer in this research to be verified (in this layer, the algorithm of Encoding/Decoding should be selected and fully defined in the formalization and at the end, fully validate the control flow of "$\mathcal{E}_{Inv}([\ ], decrypted_m) = (R, M)$" for a given message M).
$\boxed{3}$	Skeleton of the proof is validation of the control flow of "$\mathcal{DEC}_{\mathcal{M}}\ Fp\ (\mathcal{R}\ a)\ p\ ==\ m$" which is main goal in this research to be verified.
$\boxed{2}$	Transition from Z/q to Z should be proven as a centering method (which used in decryption phases) chooses the appropriate interval, thus the polynomial a equals "$p.r \otimes g \oplus m \otimes f$" exactly, and not merely modulo q, so the $wrap_{failure}$ not be happened (in this layer, algorithm of centering method should be selected and fully defined in the formalization and at the end, fully validate control flow of it).
$\boxed{1}$	Handling the $gap_{failure}$ which happen with the probability of $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ for a given parameter set of (d, n, q) and output $a = (\mathcal{A}\ e\ f)$, so that satisfy condition $Width(a) < q$ with the chance of $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ (it may need to proof process with the operation on rat in the complex function $\mathcal{P}rob_{\mathcal{D}ecrypt\_\mathcal{F}ail(d,n,q)}$ or if we bound the failure probability, it may be required to switch to probabilistic reasoning).

VI. Conclusions

We explored the application of a formal proof system to the simplified version of NTRUEncrypt function introduced in [1]. More precisely, we proved the functional correctness of this algorithm formally with the Isabelle/HOL proof system. In this case, we proved formally that this scheme is correct what is a necessary condition for the usefulness of any cryptographic encryption scheme. A formal analysis with computer support provides a complex, but useful approach to verify the functional correctness of implementations of cryptographic algorithms. This formalization and verification is not a general scheme with high strong design to ideally underlay all further works in this area, but has some main advantages which enumerated as following:

- This research can be seen as a start point for further formal proofs in NTRUEncrypt scheme.
- Four partially independent layers introduced in this research (as a high level framework for formal verification of NTRUEncrypt scheme) which validated just for layer 3 (Table 1) in this research and other layers can be studied in further works (although the validation of layer 3 in this research can be refine in the further studies).
- The computer-proven lemma augment the given database that is basic for many Isabelle theories in the related area.
- The functions and other formal primitives which defined/declared in this research can used in the further studies.
- Difficulties and problems in the formalization and verification phases enumerated in this paper, also in the most of them, the solution idea introduced and can be resolved easily in the further studies.
- An algebraic type class hierarchy introduced in this research which can be used in the further studies.

VII. Further Studies

As mentioned in section 6 (Conclusion), difficulties and problems in the formalization/verification phases enumerated in the different sections (subsections) of this research, so that, most important of them can be resolved by following further studies:

- As the constants are not polymorphic, the generality can equally be expressed inside HOL, e.g., by turning all unspecified constants into parameters of a locale and the resulting development will more usable, because the parameters can be instantiated inside the logic.
- Ideally, the algorithms in this research should be defined (rather than axiomatised as is done in this research).
- The important properties in this research should be proven under as weak assumptions as possible.
- This research restrict the parameters of the encryption algorithm to a small set of values, so generality of this parameter set can be studied to have more usefulness of the formal proof.
- Layer 1 of the formal proof (Table 1) which corresponded to Handling the $gap_{failure}$, can be studied to be verified (as far as possible) in the further studies.
- Layer 2 of the formal proof (Table 1) which corresponded to centering method functionality for handling $wrap_{failure}$, should be studied to be verified (as far as possible) in the further studies.
- Layer 4 of the formal proof (Table 1) which corresponded to Encoding/Decoding functions can be studied to be verified (as far as possible) in the further studies.
- A lot of space in this research is spent on re-inventing existing libraries (in particular, the algebraic type class hierarchy and vectors of fixed length) to have short and elegant proofs (that are not possible easily with the existing libraries), thus the benefits from all the corollaries and the setup for proof automation that has been developed for the existing primitives in Isabelle/HOL libraries be lost and also extending the area of this proof to be used in other external theorems using the standard

Isabelle/HOL libraries, faced with problems, so use the existing libraries as far as possible in the further studies.

- This research partially focus on elegance or encapsulating fundamental insights that are adaptable and reusable so these properties should be consider more in further studies.

REFERENCES

[1] Hoffstein, J., Pipher, J., Silverman, J.H., "NTRU: A ring-based public key cryptosystem", *Lecture Notes in Computer Science*, 1998.

[2] Bernstein, D.J., "Post-Quantum Cryptography", Department of Computer Science, University of Illinois at Chicago, 2008.

[3] Silverman, J.H., Whyte, W., "Estimating decryption failure probabilities for NTRUEncrypt", *Technical report*, NTRU Cryptosystems, Report #018, version 1, available at http://www.ntru.com, 2003.

[4] Kaiser, M., Buchmann, J., "Formal Analysis of a Public-Key Algorithm", *International Journal of Computer Science 2.2*, 2007.

[5] Kusch, S., Buchmann, J., "Formalizing the DSA Signature Scheme in Isabelle/ HOL", Diplomarbeit, Technische Universit ̈at Darmstadt, 2007.

[6] Kusch, S., Kaiser, M., "A Computer Proven Application of the Discrete Logarithm Problem", *International Journal of Computer Science 2.2*, 2007.

[7] Nipkow, T., Paulson, L.C., Wenzel, M., "Isabelle/HOL—a proof assistant for higher-order logic", *Lecture Notes in Computer Science*, vol. 2283. Springer, 2002. doi:10.1007/3-540-45949-9.

[8] Nipkow, T., Klein, G., "Concrete Semantics-A Proof Assistant Approach", Springer, 2014.

[9] Nipkow, T., "Programming and Proving in Isabelle/HOL", http://isabelle.informatik.tu-muenchen.de/, 2012.

[10] Wenzel, M., "The Isabelle/Isar Reference Manual", TU MÄunchen, MÄunchen, 1999, http://isabelle.in.tum.de/doc/isar-ref.pdf.

[11] Holzl, J., "Proving Real-Valued Inequalities by Computation in Isabelle/HOL", Diploma thesis, Institut f ̈ur Informatik, Technische Universit ̈at M ̈unchen, 2009.

[12] "IEEE P1363.1. Public-key cryptographic techniques based on hard problems over lattices", http://grouper.ieee.org/groups/1363/lattPK/index.html, Accessed May 31, 2014.

[13] Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W., "Hybrid lattice reduction and meet in the middle resistant parameter selection for ntruencrypt", NTRU Cryptosystems, Inc., Acton, MA, Tech. Rep., 2007.

[14] Ballarin, C., "Tutorial to locales and locale interpretation", In L. Lamb án, A. Romero, and J. Rubio, editors, Contribuciones Cient ficas en honor de Mirian Andr és G ómez. Servicio de Publicaciones de la Universidad de La Rioja, Logro ño, Spain, 2010.

[15] Mallagaray, J.D., "Proofs of properties of finite-dimensional vector spaces using Isabelle/HOL", Universidad de La Rioja, 2011/2012, http://www.unirioja.es/cu/jodivaso/degree_thesis/.

[16] Security Innovation, https://www.securityinnovation.com/. Accessed August 9, 2013.

[17] Nipkow, T., "What's in Main", Isabelle TUM, 2013, http://isabelle.in.tum.de/doc/main.pdf.

[18] Krauss, A., "Defining Recursive Functions in Isabelle/HOL", Isabelle TUM, 2008, http://isabelle.in.tum.de/documentation.html.

[19] Sternagel, C., Thiemann, R., "Executable matrix operations on matrices of arbitrary dimensions", *Archive of Formal Proofs*, 2010.

[20] Lindenberg, C., Wirt, K., Buchmann, J., "Formal proof for the correctness of RSA-PSS", *IACR Cryptology ePrint Archive* 2006, 11.

[21] Silverman, J.H., "An Introduction to the Theory of Lattices and Applications to Cryptography", Computational Number Theory and Applications to Cryptography, University of Wyoming, 2006.

[22] Isabelle-TUM, https://isabelle.in.tum.de/(2013). Accessed December 05, 2013.

[23] Appel, A.W., "Verification of a Cryptographic Primitive: SHA-256", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Princeton University, 2015.

Authors' Profiles

Gholam Reza Moghissi, is currently a Master student (Information Security) in ICT Department at Malek-e-Ashtar University of Technology, Tehran, Iran.

Ali Payandeh, is now an assistant professor in the Department of Information and Communications Technology at Malek-e-Ashtar University of Technology, Iran.

How to cite this paper: Gholam Reza Moghissi, Ali Payandeh,"Formal Verification of NTRUEncrypt Scheme", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.4, pp.44-55, 2016.DOI: 10.5815/ijcnis.2016.04.06