Internet Engineering Task Force
Internet-Draft
Intended Status: Informational
Expires: January 19, 2018

C. Tjhai
M. Tomlinson
A. Cheng
Post-Quantum
G. Bartlett
Cisco Systems
July 18, 2017

Hybrid Quantum-Safe Key Exchange for Internet Key Exchange Protocol Version 2 (IKEv2) draft-tjhai-ipsecme-hybrid-qske-ikev2-00

Abstract

This document describes the optional key-exchange payload of Internet Key Exchange Protocol Version 2 (IKEv2) that carries quantum-safe key exchange data. This optional payload is used in conjunction with the existing Diffie-Hellman key exchange to establish a quantum-safe shared secret between an initiator and a responder. The optional payload supports a number of quantum-safe key exchange schemes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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 December 21, 2017.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Description

Internet Key Exchange Protocol (IKEv2) as specified in RFC 7296 [RFC7296] uses the Diffie-Hellman algorithm [DH] to establish a shared secret between an initiator and a responder. The security of the Diffie-Hellman algorithm relies on the difficulty to solve a discrete logarithm problem when the order of the group parameter is large enough. While solving such a problem remains difficult with current computing power, it is believed that general purpose quantum computers can easily crack this problem, implying that the security of IKEv2 is compromised. There are, however, a number of

cryptosystems that are conjectured to be resistant against quantum computer attack.

1.2. Proposed Extension

This document describes a method to extend IKEv2, whilst maintaining backwards compatibility, to perform key exchange that is robust against quantum computers. The idea is to use an optional key exchange payload using a quantum-safe key exchange algorithm, in addition to the existing Diffie-Hellman key exchange. The secrets established from each key exchange are combined in a way such that should the quantum-safe secret not be present, the derived shared secret is equivalent to that of the standard IKEv2; on the other hand, a quantum-safe shared secret is obtained if both key exchange payloads are present. This extension also applies to key exchanges in IKE Security Associations (SAs) for Encapsulating Security Payload (ESP) [ESP] or Authentication Header (AH) [AH], i.e. Child SAs, in order to provide a stronger guarantee of forward security.

The goals of this extension are:

- to allow an additional key exchange using a quantum-safe algorithm to be used alongside the existing key exchange algorithm while we are transitioning to a post-quantum era;
- to keep the modifications to IKEv2 to a minimum whilst maintaining compatibility with IKEv2; and
- to provide a path to phase out the existing Diffie-Hellman key exchange in the future.

It is expected that implementers of this specification are familiar with IKEv2 [RFC7296], and are knowledgeable about quantum-safe cryptosystems, in particular key exchange mechanisms and key encapsulation mechanisms instantiated with public-key encryption.

The remainder of this document is organized as follows. Subsection 1.3 provides an overview of the terminology and the abbreviations used in this document. Section 2 specifies how quantum-safe key exchange is performed between two IKE peers and how keying materials are derived in both IKE and Child SAs. The rationale behind the approach of this extension is described in Section 3. Section 4 discusses security considerations. Section 5 describes IANA considerations for the name spaces introduced in this document. This is followed by a list of cited references and the authors' contact information.

1.3. Terminology

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [RFC2119]. In addition to using the terms defined in IKEv2 [RFC7296], this document uses the following list of abbreviations:

KEM: It stands for key encapsulation mechanism whereby key material is transported using a public-key algorithm.

QSKE: Denotes a quantum-safe key exchange payload, which is similar to Key Exchange (KE) payload.

QSSS: Denotes a quantum-safe shared secret (QSSS) established from QSKEi and QSKEr payloads. This entity is similar to the Diffie-Hellman shared secret g^ir as defined in RFC 7296.

Q-S Group: It stands for Quantum-Safe Group and it represents a quantum-safe cryptography algorithm for key exchange. Each group corresponds to an algorithm with a specific set of parameters.

2. Hybrid Quantum-Safe Key Exchange

IKEv2 key exchange occurs in IKE_SA_INIT or CREATE_CHILD_SA message pair which contains various payloads for negotiating cryptographic algorithms, exchanging nonces, and performing a Diffie-Hellman shared secret exchange for an IKE SA or a Child SA. These payloads are chained together forming a linked-list and this flexible structure allows an additional key exchange payload, denoted QSKE, to be introduced. The additional key exchange uses algorithms that are currently considered to be resistant to quantum computer attacks. These algorithms are collectively referred to as quantum-safe algorithms in this document.

2.1. Quantum-Safe Group Transform Type

In generating keying materials within IKEv2, both initiator and responder negotiate up to four cryptographic algorithms in the SA payload of an IKE_SA_INIT or a CREATE_CHILD_SA exchange. One of the negotiated algorithms is an ephemeral Diffie-Hellman algorithm, which is used for key-exchange. This negotiation is facilitated by the Transform Type 4 (Diffie-Hellman Group) where each Diffie-Hellman group is assigned a unique Transform ID.

In order to enable a quantum-safe key exchange in IKEv2, the various quantum-safe algorithms MUST be negotiated between two IKEv2 peers. Transform Type #tba (Quantum-Safe Group) is used to facilitate this

negotiation. It is identical to Transform Type 4, except that the latter deals with various Diffie-Hellman groups only whereas the former handles quantum-safe algorithms only. Each quantum-safe algorithm is assigned a unique Transform ID.

Whilst all the key exchange algorithms in Transform Type 4 are based on Diffie-Hellman, some of the algorithms in Transform Type #tba are Diffie-Hellman-like, and the rest of the algorithms use key-encapsulation-mechanism (KEM). In the case of KEM, the initiator randomly generates a random, ephemeral public and private key pair, and sends the public key to the responder in QSKEi payload. The responder generates a random entity, encrypts it using the received public key, and sends the encrypted quantity to the initiator in QSKEr payload. The initiator decrypts the encrypted payload using the private key. After this point of the exchange, both initiator and responder have the same random entity from which the quantum-safe shared secret (QSSS) is derived.

The Transform Type #tba (Quantum-Safe Group) is defined as an optional type in IKE, AH and ESP protocols. This transform type MUST NOT exist if there is no Transform Type 4 in a proposal.

For Transform Type #tba, the defined list of quantum-safe Transform IDs are listed below. Note that the values below are only current as of the publication date of this document. Readers should refer to [IKEV2IANA] for the latest values.

Name	Number	Key exchange
RLWE 128	1	Diffie-Hellman-like
NewHope 128	2	Diffie-Hellman-like
NTRU EES743EP1	3	KEM
NTRU-Prime 216	4	KEM

2.2. IKE_SA_INIT Exchange

The IKE_SA_INIT request and response pairs negotiate cryptographic algorithms, exchange nonces and perform a key exchange for an IKE SA.

```
Initiator                          Responder
-------------------------------------------------------------
HDR, SAi1, KEi, [QSKEi,]
     Ni                      -->
```

The initiator sends a QSKEi payload which contains parameters needed to established a quantum-safe shared secret. The QSKEi payload is marked as OPTIONAL so that it will be ignored by a responder who does not understand it. In this particular case, the responder will

respond with a set of payloads as defined in IKEv2 [RFC7296], and therefore maintaining compatibility with existing implementation. On the other hand, if the responder implements this specification, it will respond as follows:

```
                                <--  HDR, SAr1, KEr, [QSKEr,]
                                         Nr, [CERTREQ]
```

The QSKEr payload completes the quantum-safe shared secret between the initiator and responder.

At this point in the negotiation, both initiator and responder is able to compute:

- a shared Diffie-Hellman secret from KEi and KEr pair, and
- a quantum-safe shared secret from QSKEi and QSKEr pair.

Using these two shared secrets, each peer generates SKEYSEED, from which all keying materials for protection of the IKE SA are derived. The quantity SKEYSEED is computed as follows:

```
SKEYSEED = prf(Ni | Nr, g^ir | QSSS)
```

where prf, Ni, Nr, and g^ir are defined as in IKEv2 [RFC7296]. QSSS is represented as an octet string. The seven secrets derived from SKEYSEED, namely SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr, are generated as defined in IKEv2 [RFC7296].

Because the initiator sends a QSKE payload, which contains quantum-safe data, in the IKE_SA_INIT, it must guess a Q-S group that the responder will select from its list of proposed groups. If the initiator guesses incorrectly, the responder will respond with a Notify payload of type INVALID_QSKE_PAYLOAD indicating the selected Q-S group and the initiator MUST retry the IKE_SA_INIT with the corrected Q-S group. There are two octets of data associated with this notification, which contains the accepted Quantum-Safe Group Transform Type number in big endian order. As in the case of INVALID_KE_PAYLOAD, the initiator MUST again propose its full set of acceptable cryptographic suites because the rejection message was not authenticated, which may lead to any potential vulnerabilities exploitation.

2.3. CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA exchange is used to create new Child SAs and to rekey both IKE SAs and Child SAs. If the CREATE_CHILD_SA request contains a KE payload, it MAY also contain an optional QSKE payload

to enable quantum-safe forward secrecy for the Child SA. The keying material for the Child SA is a function of Sk_d established during the establishment of the IKE SA, the nonces exchanged during the CREATE_CHILD_SA exchange, the Diffie-Hellman value, and the quantum-safe data (if QSKE payload is included in the CREATE_CHILD_SA exchange).

If a CREATE_CHILD_SA request includes a QSKEi payload, at least one of the SA offers MUST include a Q-S group in one of its transform structures. The Q-S group MUST be an element of the group that the initiator expects the responder to accept. If the responder selects a different Q-S group, the responder MUST reject the request by sending INVALID_QSKE_PAYLOAD Notify payload. The responder's preferred Q-S group is indicated in this notify payload. In the case of a rejection, the initiator should retry with another CREATE_CHILD_SA request containing a Q-S group that was indicated in the INVALID_QSKE_PAYLOAD Notify payload.

2.3.1. New Child SAs from the CREATE_CHILD_SA Exchange

The CREATED_CHILD_SA request and response pair to create a new Child SA is shown below:

```
   Initiator                          Responder
   -------------------------------------------------------------
   HDR, SK {SA, Ni,
      [KEi,] [QSKEi,] TSi, TSr} -->

                                 <--  HDR, SK {SA, Nr,
                                         [KEr,] [QSKEr,] TSi, TSr}
```

The initiator sends an encrypted request containing SA offer(s), a nonce, optional Diffie-Hellman and quantum-safe key exchange data and the proposed Traffic Selectors.

The responder replies with an encrypted response containing the accepted SA offer, a nonce, a Diffie-Hellman value if KEi was included in the request and the expected Diffie-Hellman group was selected, a quantum-safe data if QSKEi was included in the request and the expected Q-S group was selected, and the accepted Traffic Selectors.

The keying material of these CREATE_CHILD_SA exchanges that have both KE and QSKE payloads is defined as:

```
   KEYMAT = prf+(SK_d, QSSS (new) | g^ir (new) | Ni | Nr)
```

where prf+, Sk_d, g^ir (new), Ni and Nr are defined in IKEv2

[RFC7296], and QSSS (new) is the shared secret from the ephemeral quantum-safe key exchange. The QSSS quantity is represented as an octet string.

2.3.2. Rekeying IKE SAs with the CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA request and response pair for rekeying an IKE SA is shown below:

```
   Initiator                          Responder
   -----------------------------------------------------------
   HDR, SK{SA, Ni,
          KEi[, QSKEi]}     -->
                            <--      HDR, SK {SA, Nr,
                                         KEr[, QSKEr]}
```

The initiator sends an encrypted request containing amongst other payloads, a KEi payload which carries a Diffie-Hellman value, and an OPTIONAL QSKEi payload which carries a quantum-safe data.

The responder replies with an encrypted response containing a number of payloads. If the responder selects a Diffie-Hellman group that matches one of the proposed group(s), a KEr payload containing a Diffie-Hellman public value is replied in the encrypted response. If the request contains a QSKEr payload and the responder selects a Q-S group that matches one of the proposed group(s), a QSKEr payload containing quantum-safe data is sent in the reply.

The quantity SKEYSEED for the new IKE SA is computed as follows:

```
   SKEYSEED = prf(SK_d (old), QSSS (new) | g^ir (new) | Ni | Nr)
```

where prf, SK_d (old), g^ir (new), Ni and Nr are defined in IKEv2 [RFC7296], QSSS (new) is the shared secret from the ephemeral quantum-safe key exchange. The QSSS quantity is represented as an octet string.

2.3.3. Rekeying Child SAs with the CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA request and response pair for rekeying a Child SA is shown below:

```
   Initiator                          Responder
   -----------------------------------------------------------
   HDR, SK {N(REKEY_SA), SA,
      Ni, [KEi,] [QSKEi,]
      TSi, TSr}              -->
```

```
                              <--    HDR, SK {SA, Nr,
                                       [KEr,] [QSKEr,] TSi, TSr}
```

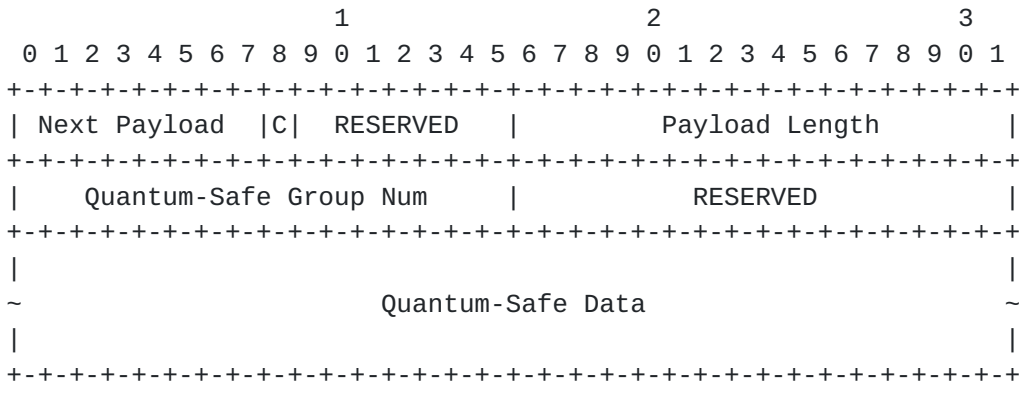
Both KEi and QSKEi payloads are OPTIONAL. The KEi and QSKEi payloads, which are sent encrypted by the initiator, carry a Diffie-Hellman value and quantum-safe data respectively.

If the CREATE_CHILD_SA request includes KEi and QSKEi payloads, provided that a Diffie-Hellman group and a Q-S group are present in the SA offers, the responder replies with an encrypted response containing both KEr and QSKEr payloads.

The keying material computation of this exchange is the same as that defined in [Section 2.3.1].

2.4. QSKE Payload Format

The quantum-safe key exchange payload, denoted QSKE in this document, is used to exchange a quantum-safe shared secret between two IKE peers. The QSKE payload consists of the IKE generic payload header, a two-octet value denoting the Quantum-Safe Group number, and followed by the quantum-safe data itself. The format of the QSKE payload is shown below.



The length of the quantum-safe data varies depending on the type of quantum-safe cipher. The content type of quantum-safe data is also dependent on the type of quantum-safe cipher. For quantum-safe ciphers that use Diffie-Hellman-like key exchange, the content of the quantum-safe data is the proposed/accepted cipher's public value. For ciphers that use KEM, the content is either a random public-key of the proposed quantum-safe cipher in the case of QSKEi payload, or the content is a ciphertext produced using the received public-key in the case of QSKEr payload.

The Quantum-Safe Group Num identifies the quantum-safe cipher with which the quantum-safe data was computed. The Quantum-Safe Group Num

MUST match the Q-S group specified in a proposal in the SA payload sent in the same message. If the proposal in the SA payload does not specify a quantum-safe cipher, the QSKE payload MUST NOT be present. If the responder selects a Q-S group that does not match the proposed group, the quantum-safe key exchange MUST be rejected with a Notify payload of type INVALID_QSKE_PAYLOAD. The chosen Q-S group is indicated in the INVALID_QSKE_PAYLOAD Notify payload and the initiator can restart the exchange with that group.

The payload type for the QSKE payload is TBA (TBA).

3. Design Rationale

In general, the size of QSKE payload is larger than that of the KE counterpart and sending it in the IKE_SA_INIT may prevent peers from establishing IPSec Security Association (SA) due to fragmentation. While the fragmentation issue may be addressed by sending QSKE in the IKE_AUTH exchange, it is decided that QSKE should still be exchanged in the IKE_SA_INIT. The rationale behind this decision is discussed below.

3.1. Threat Categories

The treats to the IKE exchange can be broken into two categories:

1. From current day until general purpose quantum computers are available.

   The addition of the QSKE allows the IKEv2 exchange to be secured against an adversary who captures all control plane (IKE) and data plane (ESP) traffic, with the intention of breaking the IKE exchange (when quantum computers become available) and subsequently being able to view the data plane traffic. The use of the QSKE in the IKE_SA_INIT results in the IKE SA becoming quantum secure against future attacks.

2. After general purpose quantum computers are available.

   Once general purpose quantum computers are available there are two types of attack:

   - Active attack

     Assuming that a general purpose quantum computer is available and an adversary can manipulate the IKE exchange in real time. The attacker can break Diffie-Hellman in real time, but not the QSKE. This results in the IKE_AUTH exchange being secure as the QSKE is included in the

derivation of key material used to secure the IKE_AUTH exchange.

However, an active attacker who can sit between two hosts and impersonate each host can perform a man-in-the-middle (MitM) attack when the authentication method is not quantum secure. This includes any asymmetric authentication method and non-quantum computer resistant Extensible Authentication Protocol (EAP) authentication. For authentication methods which are quantum secure, such as using shared key message integrity code comprising a shared-secret with sufficient entropy (256 bits), this allows for the IKEv2 exchange to be secured against an active adversary when including the QSKE.

- Passive attack

  As per the first category, the addition of the QSKE allows the IKEv2 exchange to be secured against an adversary who captures all control plane (IKE) and data plane (ESP) traffic, with the intention of breaking the IKE exchange.

3.2. Dealing with Fragmentation

In some instances, the QSKE public value will be large enough to cause fragmentation to occur at the IP layer. In practice, there will be cases where IKE traffic fragmented at the IP layer will be dropped by network devices such as NAT/PAT gateways, Intrusion Prevention System (IPS), firewalls and proxies, that cannot handle IP fragments or are configured to block IP fragments. This blocked traffic will prevent the IKE session from being established. The issue with fragmentation can easily be avoided by moving the QSKE to the IKE_AUTH exchange and by employing IKEv2 Message Fragmentation [RFC7383]. The implication of this is that while all the Child SAs, which carry the data traffic, would be quantum secure, the IKE SA itself would not be, resulting in the disclosure of IKE identities and IPsec proxies. Furthermore by sending the QSKE in IKE_AUTH and not IKE_SA_INIT would allow an active attacker with a quantum computer to perform attacks against IKE such as forging an identity used for authentication, abuse of attributes sent in the CFG exchange, MitM attack, DoS, etc. It is believed that the trade off to deliver a quantum resistant IKE SA is of greater security benefit than the issues that could be encountered due to fragmentation at the IP layer. It is worth noting that encapsulating IKE traffic within TCP [IKETCPENCAP] is a simple method to prevent IKE_SA_INIT traffic being fragmented at the IP layer.

The following table gives an idea of the common size of the QSKE payload in the proposed schemes.

Scheme	QSKE size (octets)
RLWE 128	4096
NewHope 128	1792
NTRU EES743EP1	1030
NTRU-Prime 216	1200

It is evident that both NewHope 128 and RLWE 128 will naturally increase an IP Maximum Transmission Unit (MTU) to be larger than 1500 octets which is common for most Internet traffic, resulting in the IKE_SA_INIT being fragmented at the IP layer.

3.3. Removal of the Diffie-Hellman exchange

The IKE_SA_INIT exchange currently mandates the use of the Diffie-Hellman. As the Diffie-Hellman exchange is not quantum secure and the QSKE exchange is quantum secure, the addition of the QSKE can be thought of making the Diffie-Hellman redundant. This draft does not advise removing the use of Diffie-Hellman, though future implementations that have migrated to using QSKE could remove the requirement to send the Diffie-Hellman exchange with the QSKE providing the same functionality. Sending the QSKE in the IKE_SA_INIT allows for a simple transition to only using QSKE should the need to remove the Diffie-Hellman exchange occur.

4. Security Considerations

The key length of the Encryption Algorithm (Transform Type 1), the Pseudorandom Function (Transform Type 2) and the Integrity Algorithm (Transform Type 3), all have to be of sufficient length to prevent attacks using Grover's algorithm [GROVER]. In order to use the extension proposed in this document, the key lengths of these transforms SHALL be at least 256 bits long in order to prevent any quantum attacks from succeeding. Accordingly the post-quantum security level achieved is at least 128 bits.

The quantities SKEYSEED and KEYMAT are calculated from shared secrets, g^ir and QSSS, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of g^ir, the quantity QSSS ensures that neither SKEYSEED nor KEYMAT is compromised. This assumes that the algorithm defined in the Transform Type 2 is quantum-safe.

Because some quantum-safe public values are in the order of several

KB, a IKEv2 message that contains such a QSKE payload will exceed the path Maximum Transmission Unit (MTU) and the message may be fragmented at the IP level. This presents the possibility of an attack vector that relies on IP fragmentation. One such attack vector is to mount a denial of service by swamping a receiver with IP fragments [DOSUDPPROT]. This issue could be mitigated by employing TCP encapsulation [IKETCPENCAP].

The authenticity of the SAs established under IKEv2 is protected using a pre-shared key, RSA, DSS, or ECDSA algorithms. Whilst the pre-shared key option, provided the key is long enough, is quantum-safe, the other algorithms are not. Moreover, in implementations where scalability is a requirement, the pre-shared key method may not be suitable. Quantum-safe authenticity may be provided by using a quantum-safe digital signature and several quantum-safe digital signature methods are being explored by IETF. For example the hash based method, XMSS has the status of an Internet Draft, see [XMSS]. Currently, quantum-safe authentication methods are not specified in this document, but are planned to be incorporated in due course.

It should be noted that the purpose of quantum-safe algorithms is to prevent attacks, mounted in the future, from succeeding. The current threat is that encrypted sessions may be subject to eavesdropping and archived with decryption by quantum computers taking place at some point in the future. Until quantum computers become available there is no point in attacking the authenticity of a connection because there are no possibilities for exploitation. These only occur at the time of the connection, for example by mounting a MitM attack. Consequently there is not such a pressing need for quantum-safe authenticity.

The use of the QSKE provides an method for malicious parties to send IKE_SA_INIT initiator messages containing QSKE of type KEM and with random values. As the standard behavior is for the responder to generate a random entity, encrypt it using the received public key (which would be a random value), and sends the encrypted quantity to the initiator in QSKEr payload. This allows for a simply method for malicious parties to cause a VPN gateway to perform excessive processing. To mitigate against this threat, implementations can make use of the COOKIE notification as defined in [RFC7296], to mitigate spoofed traffic and [RFC8019] to minimize the impact from hosts who use their own IP address.

5. IANA Considerations

This document defines a new IANA registry for IKEv2 Transform Types.

Description	Trans. Type	Used In
Quantum-Safe Group (Q-S)	tba	Optional in IKE, AH & ESP

A number of Transform IDs of the Q-S group Transform Type are also defined. The initial values are listed below:

Name	Value
RLWE 128	1
NewHope 128	2
NTRU EES743EP1	3
NTRU-Prime 216	4

In order to transport quantum-safe data to establish a quantum-safe SA, this extension registers a new key exchange payload in the IKEv2 Payload Types of the IANA registry:

Description	Notation	Value
QSKE Payload	QSKE	tba

This extension also specifies a new error type in the IKEv2 Notify Message Types - Error Types of the IANA registry:

Error Type	Value
INVALID_QSKE_PAYLOAD	tba

6. References

[ADPS] Alkim, E., Ducas, L., Poppelmann, T., and Schwabe, P., "Post-quantum Key Exchange - a New Hope", 25th USENIX Security Symposium, pp. 327-343, 2016.

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[BCNS15] Bos, J., Costello, C., Naehrig, M., and Stebila, D., "Post-quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem", IEEE Symposium on Security and Privacy, pp. 553-570, 2015.

[DH] Diffie, W., and Hellman, M., "New Directions in Cryptography", IEEE Transactions on Information Theory,

V.IT-22 n. 6, June 1977.

[DOSUDPPROT]
Kaufman, C., Perlman, R., and Sommerfeld, B., "DoS protection for UDP-based protocols", ACM Conference on Computer and Communications Security, October 2003.

[ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[GROVER] Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), 1996

[IKETCPENCAP]
Pauly, T., Touati, S., and Mantha, R., "TCP Encapsulation of IKE and IPsec Packets", draft RFC, May 2017, <https://tools.ietf.org/html/draft-ietf-ipsecme-tcp-encaps-10>.

[IKEV2IANA]
IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters/>.

[LOGJAM] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J., Heninger, N., Springall, D., Thome, E., Valenta, L., VanderSloot, B., Wustrow, E., Beguelin, S., and Zimmermann, P., "Imperfect forward secrecy: How Diffie-Hellman fails in practice", Proc. 22rd ACM SIGSAC Conference on Computer and Communications Security, pp. 5-17, 2015.

[NTRU] Hoffstein, J., Pipher, J., and Silverman, J., "NTRU: A Ring-Based Public Key Cryptosystem", Lecture Notes in Computer Science, pp. 267-288, 1998.

[NTRUPRIME]
Bernstein, D., Chuengsatiansup, C., Lange, T., and van Vredendaal, C., "NTRU Prime", IACR Cryptology ePrint Archive: Report 2016/461, 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and Kivinen, T., "Internet Key Exchange Protocol Version 2

(IKEv2)", RFC 7296, October 2014.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, November 2014.

[RFC8019] Nir, Y., Smyslov, V., "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, November 2016.

[XMSS] Huelsing, A., Butin, D., Gazdag, S., and Mohaisen, A., "XMSS: Extended Hash-Based Signatures", Crypto Forum Research Group Internet Draft, 2017

Appendix A. Quantum-safe Ciphers

Each of the specific quantum-safe ciphers is assigned a unique Transform ID. All of the selected quantum-safe ciphers are based on lattice construction. Specifically the ciphers fall into the categories of Ring Learning With Errors, NTRU and Streamlined NTRU Prime. In each case the selected parameters are chosen so as to achieve at least 128 bits of post-quantum security.

Appendix A.1. Ring Learning With Errors

Ring Learning with Errors is a cryptographic primitive that relies on the worst-case hardness of a shortest vector problem in ideal lattices. It is commonly abbreviated as RLWE. The security parameters are given by an integer n which is a power of 2, a prime integer q, an array of n coefficients denoted by {a} and a standard deviation sigma along with the type of error distribution X. Note that each coefficient of {a} is less than the prime q and is sampled from distribution X. Let a(x) be a polynomial, whose coefficients are given by {a}, the RLWE problem can be stated as follows: given polynomials a(x), b(x) and a small polynomial e(x), find the secret s(x) from the relationship a(x) * s(x) + b(x) = e(x) modulo q.

RLWE 128

This set of parameters follows the system described by Bos et al [BCNS15]. Using a fixed coefficient array {a} in this way may result in security vulnerabilities such as "all-for-the-price-of-one" precomputation attacks such as the Logjam attack on the classical Diffie-Hellman key exchange [LOGJAM]. As has been pointed out since, this is straightforwardly solved by the coefficient array {a} being generated on-the-fly for each key exchange from a seed value shared

by the initiator and responder. The fixed coefficient array {a} is also avoided in similar fashion in NewHope 128 (see below).

The set of parameters that is proposed by Bos et al is given as follows:

```
n = 1024
q = 2^32 -1
sigma = 8/sqrt(2 * PI)
X = discrete Gaussian
{a} = 29FE0191, DD1A457D, 3534EE4B, 6450ED74, BBFE9F64, 92BF0F31,
      8DCF8995, 4C5E30D0, 9E2ED04D, 8C18FE0B, 1A70F2E7, 2625CD93,
      0065DA14, 6E009722, E6A70E8B, AEF6EF56, 8C6C06AF, 9E59E953,
      4995F67B, E918EE9D, 8B4F41A7, 0D811041, F5FE6458, 3C02B584,
      CBCFC8FD, 5A01F116, 73408361, 44D3A098, BBDEECF6, 90E09082,
      F8538BA4, F9600091, D8D30FEF, 56201487, ACB2159D, 38F47F77,
      ED7A864F, 8FC785CA, 7CBD6108, 3CA577DE, FF44CCC2, A1385A79,
      5C88E3AD, 177C46A9, DA4A4DD8, 2AA3594F, A4A5E629, 47CA6F6E,
      B2DF1BC6, 6841B78E, 0823F5A8, A18C7D52, 7634A0D1, DA1751BA,
      18B9D25D, 5B2643BC, ACC6975D, 48E786F4, 05E3ED4E, 4DC86568,
      3F5C5F99, 585DBFD7, EF6E0715, 7D36B823, 12D872CD, D7B78F27,
      DD672BF5, 2DC7C7EB, A3033801, 50E48348, 9162A260, 0BE8F15B,
      ABB563EC, 06624C5A, 812BF7BC, 8637AC35, F44504F3, FF8577AB,
      4A0161B0, 000AEB0E, 311204AF, 2A76831B, 4D903F3A, 97204FA9,
      9EB524E3, 1757AFAC, BA369FEC, CD8F198D, 6B33C246, 51C13FCE,
      B58ACC4E, 39ACF8DA, 7BB7EBF7, EDC1449D, C7B47FDB, 9C39148D,
      4E688D7B, FAD0C2C2, 296CE85C, 6045C89C, 6441C0C6, 50C7C83A,
      C11764DD, 58D7EEA2, E57B9D0E, 4E142770, B8BFBB59, E143EBAA,
      FF60C855, 238727F0, E35B4A5B, 8F96940B, 4498A6BA, 5911093A,
      394DD002, 521B00D2, 140BDAF9, EAB67207, 21E631A6, A04AADA9,
      A96A9843, 4B44CC9B, E4D24C33, C7E7AE78, E45A6C72, CBE61D3C,
      CE5A4869, 10442A52, DB11F194, 39FC415D, 7E7BDB76, AE9EFA22,
      25F4F262, 472DD0A7, 42EBD7A0, E8038ECE, D3DB002A, 8416D2EC,
      DF88C989, 7FEA22D5, C7A3F6FE, 37409982, F45B75E2, 9A4AC289,
      90406FD6, EA1C74A5, 5777B39F, D07F1FA3, CE6EDA0D, D150ECFB,
      BEFF71BA, 50129EFC, 51CE65B9, B9FB0AB8, 770C59CB, 11F2354F,
      8623D4BB, D6FCAFD6, B2B1697C, 0D7067E2, 2BA5AFB9, D369C585,
      5B5E156C, D8C81E6E, 80CFDF16, F6F441EB, C173BAF5, 78099E3A,
      D38F027B, 4AC8D518, 8D0108A1, E442B0F1, 56F9EA3C, D0D6BBCA,
      4E17DCB4, 69BF743B, 0CCE779F, D5E59851, 63861EA2, B1CB22C1,
      BBFD2ACE, DDA390D1, EDF1059F, 04F80F89, B13AF849, 58C66009,
      E0D781C0, 588DC348, A305669D, 0D7AF67F, 32BC3C38, D725EFBA,
      DC3D9434, 22BD7ED8, 2DFD2926, 4BDEAD3A, B2D5ECE6, 16B05C99,
      FEEC7104, F6CAC918, 0944C774, CE00633B, C59DA01A, 41E8E924,
      335DF501, 3049E8EE, 5B4B8AAC, C962FC91, D6BB22B3, 0AC870EB,
      C3D99400, A0CEAC28, AF07DE1E, 831C2824, 258C5DDC, 779417E6,
      41CB33D0, 4E51076A, D1DB6038, 9E0B1C41, A9A1F90D, F27E7705,
      75892711, 5D9F1175, 85CC508B, 5CA415BE, 1858C792, FB18632F,
```

C94111EB, 937C0D28, C2A09970, 386209D9, BBDD9787, 2473F53A,
EF7E7637, CFC8630B, 2BA3B7F8, 3C0047AD, 10D76FF7, B1D9414D,
CEB7B902, A5B543F5, 2E484905, E0233C10, D061A1F8, CED0A901,
AC373CAC, 04281F37, 3609797F, DB80964D, 7B49A74F, 7699656F,
0DCEC4BC, 0EC49C2D, F1573A4E, A3708464, 9A1E89F0, 6B26DEB6,
2329FA10, CA4F2BFF, 9E012C8E, 788C1DFD, 2C758156, 2774C544,
150A1F7D, 50156D6E, 7B675DE1, 5D634703, A7CEB801, 92733DAB,
B213C00B, 304A65B1, 8856CF8E, 7FF7DD67, D0912293, 30064297,
663D051D, 01BC31B4, 2B1700BD, 39D7D18F, 1EAD5C95, 6FB9CD8B,
A09993A6, B42071C0, 3C1F2195, 7FDF4CF8, C7565A7E, 64703D34,
14B250EF, 2FA338D2, AEE576DC, 6CCED41D, 612D0913, D0680733,
8B4DBE8A, 6FFEA3D0, 46197CA2, A77F916F, FA5D7BD6, 01E22AEB,
18E462DD, 4EC9B937, DE753212, 05113C94, 7786FBD4, FB379F71,
756CF595, EAADCFAB, BBD74C2E, 1F234AC9, 85E28AEB, 329F7878,
D48FDE09, 47A60D0A, AE95163F, 72E70995, 27F9FCBF, BDCFCC41,
334BC498, EE7931A1, DFA6AEF4, 1EC5E1BF, 6221870F, CD54AE13,
7B56EF58, 4847B490, 31640CD3, 10940E14, 556CC334, C9E9B521,
499611FF, BEC8D592, 44A7DCB7, 4AC2EABD, 7D387357, 1B76D4B6,
2EACE8C9, 52B2D2A4, 0C1F2A64, 50EF2B9A, 3B23F4F4, 8DDE415E,
F6B92D2D, 9DB0F840, E18F309D, 737B7733, F9F563C5, 3C5D4AEE,
8136B0AF, C5AC5550, 6E93DEF9, 946BCCEC, 5163A273, B5C72175,
4919EFBD, 222E9B68, 6E43D8EE, AA039B23, 913FD80D, 42206F18,
5552C01F, 35B1136D, FDC18279, 5946202B, FAAE3A37, 4C764C88,
78075D9B, 844C8BA0, CC33419E, 4B0832F6, 10D15E89, EE0DD05A,
27432AF3, E12CECA6, 60A231B3, F81F258E, E0BA44D7, 144F471B,
B4C8451E, 3705395C, E8A69794, 3C23F27E, 186D2FBA, 3DAED36B,
F04DEFF1, 0CFA7BDD, FEE45A4F, 5E9A4684, 98438C69, 5F1D921B,
7E43FD86, BD0CF049, 28F47D38, 7DF38246, 8EED8923, E524E7FC,
089BEC03, 15E3DE77, 78E8AE28, CB79A298, 9F604E2B, 3C6428F7,
DCDEABF3, 33BAF60A, BF801273, 247B0C3E, E74A8192, B45AC81D,
FC0D2ABE, F17E99F5, 412BD1C1, 75DF4247, A90FC3C0, B2A99C0E,
0D3999D7, D04543BA, 0FBC28A1, EF68C7EF, 64327F30, F11ECDBE,
4DBD312C, D71CE03A, AEFDAD34, E1CC7315, 797A865C, B9F1B1EB,
F7E68DFA, 816685B4, 9F38D44B, 366911C8, 756A7336, 696B8261,
C2FA21D2, 75085BF3, 2E5402B4, 75E6E744, EAD80B0C, 4E689F68,
7A9452C6, A5E1958A, 4B2B0A24, 97E0165E, A4539B68, F87A3096,
6543CA9D, 92A8D398, A7D7FDB4, 1EA966B3, 75B50372, 4C63A778,
34E8E033, 87C60F82, FC47303B, 8469AB86, 2DAADA50, CFBB663F,
711C9C41, E6C1C423, 8751BAA9, 861EC777, 31BCCCE1, C1333271,
06864BEE, 41B50595, D2267D30, 878BA5C5, 65267F56, 2118FB18,
A6DDD3DE, 8D309B98, 68928CB2, FAE967DC, 3CEC52D0, 9CA8404B,
AADD68A8, 3AC6B1DF, D53D67EA, 95C8D163, B5F03F1D, 3A4C28A7,
E3C4B709, B8EB7C65, E76B42A3, 25E5A217, 6B6DD2B4, BEFC5DF4,
9ACA5758, C17F14D3, B224A9D3, DE1A7C8F, 1382911B, 627A2FB9,
C66AE36E, 02CC60EF, C6800B20, 7A583C77, E1CECEE8, CA0001B4,
6A14CF16, EF45DD21, 64CAA7D5, FF3F1D95, D328C67E, C85868B1,
7FBF3FEB, 13D68388, 25373DD9, 8DE47EFB, 47912F26, 65515942,
C5ED711D, 6A368929, A2405C50, FFA9D6EB, ED39A0D4, E456B8B5,

53283330, 7837FD52, 6EE46629, CAFC9D63, B781B08F, DD61D834,
FB9ACF09, EDA4444A, BB6AA57F, AED2385C, 22C9474D, 36E90167,
E6DF6150, F1B0DA3B, C3F6800E, 966302E0, 7DB1F627, F9632186,
B4933075, 81C5C817, 878CA140, 4EDE8FED, 1AF347C1, FDEB72BA,
2DA7FF9A, B9BA3638, 2BB883F1, 474D1417, C2F474A4, 1E2CF9F3,
231CB6B0, 7E574B53, EDA8E1DA, E1ACB7BB, D1E354A6, 7C32B431,
8189991B, 25F9376A, 3FFA8782, CD9038F1, 119EDBD1, 5C571840,
3DCA350F, 83923909, 9DC3CF55, 94D79DD0, D683DE2B, ECF4316A,
0FFF48D4, 5D8076ED, 12B42C97, 2284CDB4, CB245554, 3025B4D9,
B0075F35, 43A3802E, 18332B4D, 056C4467, C597E3F7, 3F0EAF9D,
F48EBB9F, 92F62731, BDB76296, 516D4466, 226102B3, 15E38046,
A683C4E0, 6C0D1962, E20CB6CA, C90C1D70, D0FF8692, D1419690,
2D6F1081, 34782E5E, AE092CD5, 90C99193, E97C0405, EAE201DA,
631FB5AC, 279A2821, DF47BA5B, FBE587E2, 6810AD2D, C63E94BD,
9AF36B42, F14F0855, 946CE350, 7E3320E0, 34130DFF, 8C57C413,
AB0723B2, F514C743, 63694BA3, 5665D23D, 6292C0B5, 9D768323,
2F8E447C, B99A00FB, 6F8E5970, 69B3BB45, 59253E02, 1C518A02,
DD7C1232, C6416C38, 77E10340, CF6BEB9A, 006F9239, 0E99B50F,
863AD247, 75F0451A, 096E9094, E0C2B357, 7CC81E15, 222759D4,
EE5BCFD0, 050F829B, 723B8FA9, 76143C55, 3B455EAF, C2683EFD,
EE7874B4, 9BCE92F7, 6EED7461, 8E93898F, A4EBE1D0, FA4F019F,
1B0AD6DA, A39CDE2F, 27002B33, 830D478D, 3EEA937E, 572E7DA3,
4BFFA4D1, 5E53DB0B, 708D21EE, B003E23B, 12ED0756, 53CA0412,
73237D35, 438EC16B, 295177B8, C85F4EE6, B67FD3B4, 5221BC81,
D84E3094, 18C84200, 855E0795, 37BEC004, DF9FAFC9, 60BEB6CD,
8645F0C5, B1D2F1C3, ECDC4AE3, 424D17F1, 8429238C, 6155EAAB,
A17BEE21, 218D3637, 88A462CC, 8A1A031E, 3F671EA5, 9FA08639,
FF4A0F8E, 34167A7D, 1A817F54, 3215F21E, 412DD498, 57B633E7,
E8A2431F, 397BD699, 5A155288, BB3538E8, A49806D2, 49438A07,
24963568, 40414C26, E45C08D4, 61D2435B, 2F36AEDE, 6580370C,
02A56A5E, 53B18017, AF2C83FC, F4C83871, D9E5DDC3, 17B90B01,
ED4A0904, FA6DA26B, 35D9840D, A0C505E4, 3396D0B5, EC66B509,
C190E41C, 2F0CE5CF, 419C3E94, 220D42CA, 2F611F4F, 47906734,
8C2CDB17, D8658F1C, 2F6745CD, 543D0D4F, 818F0469, 380FFDAE,
F5DD91E2, AD25E46A, E7039205, A9F47165, B2114C12, CF7F626F,
54D2C9FF, E4736A36, 16DB09FC, E2B787BB, 9631709A, 72629F66,
819EBA08, 7F5D73F3, A0B0B91C, FEDFBA71, 252F14EE, F26F8FA2,
92805F94, 43650F7F, 3051124F, 72CA8EAD, 21973E34, A5B70509,
B36A41CC, C52EDE5F, F706A24E, 8AAF9F92, ADF6D99A, 23746D73,
1DA39F70, 9660FC8F, A0A8CFEB, 83D5EFCA, 0AA4A72F, EEF1B2DE,
00CFCC66, 8A145369, 6376CEDA, A3262E2E, 3367BBA8, 01488C32,
5561A2AD, 40821BF2, F0C89F61, C4FAA6B3, D843377A, 67A76555,
E8D9F1CE, 943034FF, 2BD468BD, A514D935, 50CDB19D, A09C7E9E,
6FEBEC30, B1B36CF7, CD7A30BC, 36C6FE0A, 2DF52C45, 45C9957F,
65076A79, BF783DEE, 718D37F0, 098F9117, 9A70C430, 80EB1A53,
9F2505B1, 48D10D98, B8D781E9, F2376133, ECF25B98, 5A3B0E18,
2F623537, 9F0E34A4, F1027EB6, F9B16022, BA3FEC59, EF7226FD,
9F3058AA, BB51DE0E, D5435EA0, 8A6479D5, 077708B8, 9634876A,

069A260A, 168D9E6A, 9FD18E94, 8A7ACD53, 8E5A5869, 1B6F35FD,
A968913B, C72F076B, 7DDA354C, 25B0297C, D07219D5, A66862BA,
87E8EE67, FA28809B, 55762443, 31EF4956, F4F4A511, 9A9378CB,
42ABDBDE, 7AA484B7, E8EC22ED, CADDEF61, 9D18538A, A81B923E,
9C32F92A, 6D278E58, 4CDFC716, AB64814F, F832BF1A, E2C1A36B,
20675610, E78D855A, 38332C3D, 5AE0EAD9, 2E23F22D, 3C8683C5,
A351AF89, 54720D3B, ABC6E51F, 89330C8E, 600D5650, 197EA0C6,
7D502A5D, 3A536EA7, 7DF71F32, 456FE645, 3EF5E7A2, 6664BCAF,
A9D074C2, E9D9E478, 1AE9AB77, FECE7160, C618EEEC, 771B0026,
2B54F43C, 145DA102, 1B3D7949, BB6E2D9D, DB8FDC4A, 25397EBA,
9228A6E9, 56B4C69D, 337B943C, E35B716C, F7FE89A1, 023AC20D,
033165C8, 9F13B130, C1BAFB1D, A2C42C8C, 58E4D431, E10741E6,
2547589A, 8D9EF7BD, 7E322280, F49FDDC2, BE21A094, A061178A,
34D9F13B, 694D652F, 05084A2A, 2767B991, E8536AB4, EBFADF6F,
F4C8DFAC, D9967CCA, E04BCF3F, 232B3460, 9FF6E88A, 6DF3A2B0,
0FE10E99, 7B059283, 067BFB57, 8DDA26B0, B7D6652F, 85705248,
0826240C, 5DF7F52E, 47973463, B9C22D37, 9BEB265D, 493AB6FD,
10C0FB07, 947C102A, 5FEC0608, 140E07AE, 8B330F43, 9364A649,
C9AD63EF, BE4B2475, 1A09AC77, 9E40A4B0, BA9C23E7, 7F4A798D,
E2C52D66, A26EE9E0, 8C79DCE7, DD7F1C3D, 6AE83B20, 073DBA03,
B1844D97, 16D7ED6E, 5E0DE0B1, A497D717, FA507AA2, C332649B,
21419E15, 384D9CCC, 8B915A8B, BA328FD5, F99E8016, 545725EC,
ED9840ED, 71E5D78A, 21862496, 6F858B6C, F3736AE2, 8979FC2B,
5C8122D0, 0A20EB5A, 2278AA6E, 55275E74, 22D57650, E5FFDC96,
6BA86E10, 4EC5BFCC, 05AFA305, FB7FD007, 726EA097, F6A349C4,
CB2F71E4, 08DD80BA, 892D0E23, BD2E0A55, 40AC0CD3, BFAF5688,
6E40A6A5, 6DA1BBE0, 969557A9, FB88629B, 11F845C4, 5FC91C6F,
1B0C7E79, D6946953, 27A164A0, 55D20869, 29A2182D, 406AA963,
74F40C59, 56A90570, 535AC9C6, 9521EF76, BA38759B, CD6EF76E,
F2181DB9, 7BE78DA6, F88E4115, ABA7E166, F60DC9B3, FECA1EF3,
43DF196A, CC4FC9DD, 428A8961, CF6B4560, 87B30B57, 20E7BAC5,
BFBDCCDF, F7D3F6BB, 7FC311C8, 2C7835B5, A24F6821, 6A38454C,
460E42FD, 2B6BA832, C7068C72, 28CDCE59, AE82A0B4, 25F39572,
9B6C7758, E0FE9EBA, A8F03EE1, D70B928E, 95E529D7, DD91DB86,
F912BA8C, 7F478A6A, 1F017850, 5A717E10, DAC243F9, D235F314,
4F80AAE6, A46364D8, A1E3A9E9, 495FEFB1, B9058508, 23A20999,
73D18118, CA3EEE2A, 34E1C7E2, AADBADBD.

The public key size of RLWE 128 is 4096 octets and it provides 128-bit post-quantum security, although it does not provide forward secrecy due to the way coefficient array {a} is generated. A set of open source ciphersuites has been implemented and included in OpenSSL v1.0.1f and may be found within the libcrypto module. The authors anticipate RLWE being incorporated into TLS with the RLWE 128 ciphersuites being compatible with TLS 1.3. Moreover, the ciphersuites are constant time, and therefore are not vulnerable to possible side channel attacks.

NewHope 128

This is a variation on the ring learning with errors cipher by Alkim et al [ADPS] who set out in their solution to make improvements to RLWE 128. Their main improvements are to use a random coefficient array {a} for each key exchange and to reduce the size of key exchanges by using a 14-bit modulus instead of a 32-bit modulus. Additionally they relax the requirement for the error distribution to be discrete Gaussian and substitute the easier to generate Binomial distribution and provide a security justification. The set of parameters proposed by Alkim et al is given as follows:

```
n = 1024
q = 12289
sigma = sqrt(8)
X = Binomial
```

This cipher provides 128-bit post-quantum security and has a public key size of 1792 octets. It features forward secrecy. Open source, constant time, side-channel-proof, ciphersuites are publically available.

Appendix A.2. NTRU Lattices

NTRU lattices are another variant of cryptosystems based on integer lattices. The lattices have a cyclic structure as in the case of RLWE, however the NTRU problem can be stated as follows: given a polynomial a(x), a small secret polynomial e(x) and ciphertext c(x) find the secret e(x) from the ciphertext c(x) = a(x) * s(x) + e(x), modulo some integer q, where s(x) is a small secret polynomial. In the case of NTRU-Prime 216, the transmitted secret is the small polynomial s(x) and e(x) is generated incidentally as a result of streamlined data packing of the ciphertext.

NTRU EES743EP1

The inventors of the first lattice cryptosystem by Silverman et al in 1996 have been developing their system ever since. Adoption by the crypto community has been hampered by patents and a lack of security proof. Whilst the polynomial ring of RLWE is truncated modulo (x^n + 1) where n is an integer power of 2, the operations of NTRU EES743EP1 [NTRU] are defined over the polynomial ring modulo (x^n - 1) where n is 743, a prime integer. The integer modulus q is a power of 2, and it is 2048 in NTRU EES743EP1. The QSKE public value is 1030 octets.

NTRU-Prime 216

The authors, Bernstein et al [NTRUPRIME], of this recent variant of

NTRU set out to provide increased security in the light of possible vulnerabilities in the mathematical structure of rings used in NTRU and RLWE. Accordingly a Galois field, not a ring, is used in NTRU-Prime 216 with polynomials reduced modulo (x^p - x - 1), where p is a prime integer. Specifically, while the operations of NTRU EES743EP1 are defined over the polynomial modulo (x^743 - 1), the operations of NTRU-Prime 216 [NTRUPRIME] are defined over polynomial modulo (x^743 - x - 1). The integer modulus q is also chosen to be a prime to avoid any additional mathematical structure that may be potentially exploited. NTRU-Prime 216 has q = 7541. There is another parameter, an integer t, which defines the weight of one of the public key polynomials. The value of t is chosen to be 157 in this case so as to make cryptographic failure an impossibility. Cryptographic failure is theoretically possible in some ring based systems but usually these systems are designed so that this occurs with infinitesimal probability.

NTRU-Prime 216 has been designed to minimize the public key size and key encapsulation data by using streamlined data packing and the resulting QSKE public value is 1200 octets long. The ciphertext includes a 256 bit key confirmation hash. The system achieves forward secrecy. It is a very conservative design achieving 216 bits pre-quantum security and at least 128 bits post-quantum security. Open source, constant time, side-channel-proof ciphersuites are publicly available.

Authors' Addresses

C. Tjhai
Post-Quantum
EMail: cjt@post-quantum.com

M. Tomlinson
Post-Quantum
EMail: mt@post-quantum.com

A. Cheng
Post-Quantum
EMail: ac@post-quantum.com

G. Bartlett
Cisco Systems
EMail: grbartle@cisco.com