A note on Post Quantum Onion Routing

Evgnosia-Alexandra Kelesidis

"Alexandru Ioan Cuza" University of Iasi

January 2021

Abstract

Even though the currently used encryption and signature schemes are well tested and secure in a classical computational setting, they are not quantum-resistant as Shor's work proves. Taking this into account, alternatives based on hard mathematical problems that cannot be solved using quantum methods are needed, and lattice-based cryptography offers such solutions. The well-known GGH and NTRUEncrypt encryption schemes are proven secure, but their corresponding signature schemes are flawed in their design approach. Once introducing the computationally hard problems like Ring-LWE, elegant and efficient cryptographic primitives could be built.

1 Introduction

The Onion Router [SDM04] is one of the most reliable tools used for achieving anonymity on Internet. While most services focus on encrypting the payload of their communications, and assuring the confidentiality of the shared information, they do not cover the user anonymity aspect. TOR tackles this issue by directing traffic through multiple nodes (onion routers) until it reaches its destination. The information is encrypted in multiple layers, such as each node can decrypt only the layer corresponding to its key, and it only knows the previous node in the path and learns the node it has to redirect the decrypted data to. The only nodes that know the source and the destination of the communication are the first node in the path and the last. The first node has to be a guard node, as it knows the real IP of the user. In order a router to be a guard node, it has to have a good bandwidth and maintain a high uptime for weeks. Currently, the number of onion routers used to establish a path is 3.
Each node has to communicate to a central authority (Directory Authority) information about its state (named a Descriptor). The Directory Authorities collect such Descriptors in order to establish the status of the network, by voting and obtaining a Consensus Document.
Communication is established after the following pattern: a user connects to

a Directory Authority, obtains the Consensus Document and its Onion Proxy (the TOR software) selects a path (named a circuit) by choosing three Onion Routers from the list of available nodes. Afterwards, it executes a circuit establishment protocol with the three nodes for the purpose of session key establishment. These keys will be the keys used for encrypting the information in multiple layers. The circuit establishment with the guard node is made directly, but with the second is made through the first node, and with the third through the second, such as the second and third node don't know the identity of the client.
A central part of the TOR network consists in hidden services: TOR users can offer services such as being an instant messaging server without revealing their location and their identity. This is possible due to the following process: the service advertises its existence by picking some random onion routers, named introduction points, builds circuits to them, assembles an onion service descriptor that is distributed and is used by every client that wants to access the service. The client now knows the introduction points and the service's public key and picks a random onion router to act as a rendezvous point. Next, the client sends a message to one of the introduction points by building a circuit, requesting to be sent to the service, that once receiving the message encrypted with its public key and learns the address of the rendez-vous point, to whom it sends a one-time secret to it through a circuit. Finally, the rendez-vous point acknowledges the client about the successful connection establishment, that will take place only through the rendez-vous point: the client or the server will build circuits to it and it will redirect the message to the other participant. During this entire process, both the user and the server remain anonymous.
This brief description on the structure of communication in the TOR network reveals the following pattern: the main tool used in TOR for achieving anonymity is the circuit, so the robustness of the network relies on the strength of circuit creation. Currently, the key establishment between two communicating parts (the user and a router) is achieved due to the ntor protocol, which in the TOR specifications [NTS] is described as following:

- There is used a tweakable hash function $H(x, t)$, with $t_{mac}, t_{key}, t_{verify}$ as arbitrarily-chosen tweaks.
- The computation take place in a cyclic group (the group of points of an elliptic curbe: Curve 25519 with the generator $g = 9$). An element of the curve25519 is represented as a byte string of length of 32 bytes. Also the communicate parts will embed in their message the protocol ID (*protoid*).
- Considering a router with the identity key ID, he makes the following setup: generates a onion key pair $(b, B = g^b)$ and publishes B in the Descriptor.
- The client generates an ephemeral key pair $(x, X = g^x)$ and sends the following message: (ID, B, X).

- The server generates an ephemeral key pair $(y, Y = g^y)$ and computes $secret_{input} = X^y|X^b|ID|B|X|Y|protoid$, $Key_{seed}x = H(secret_input, t_{key})$, $verify = H(secret_input, t_{verify})$,
  $auth_{input} = verify|ID|B|Y|X|protoid|"Server"$, and sends the following message: $Y, AUTH = H(auth_{input}, t_{mac})$.

- The client checks if Y is on the curve and computes $secret_{input} = Y^x|Y^b|ID|B|X|Y|protoid$, $Key_{seed}x = H(secret_input, t_{key})$, $verify = H(secret_input, t_{verify})$,
  $auth_{input} = verify|ID|B|Y|X|protoid|"Server"$ and verifies whether $AUTH = H(auth_{input}, t_{mac})$. If it does, then the shared key is $g^{bx}|g^{xy}$, after applying a key derivation function to it.

A more abstract description of ntor is given in the original paper [GSU13]. The protocol's purpose, apart from confidentiality and robustness, is one-way authentication, which in TOR context can be translated as the user's need to remain anonymous to the onion router while being assured it does communicate with the intended node.

The security of the protocol is achieved if the hash function is collision resistant and provides randomness, and in the group G the GAP Diffie-Hellman assumption holds.

As we observe, the anonymity is obtained due to hard mathematical assumptions (the discrete logarithm in a cyclic group, etc), that can be solved in a Quantum setting, which is why the TOR network in its current form is no more resistant in a Post-Quantum environment.

2 Lattice-Based Post-Quantum cryptosystems

A special class of hardness assumptions lie in the lattice environment: the problems that appear are resistant to quantum approaches. The main problems that post-quantum cryptosystems rely on are the Shortes Vector Problem, the Closest Vector Problem and the Ring Learning With Errors Problem.

2.1 Definitions

Definition 1 *Let $v_1, v_2, ..., v_n \in \mathbb{R}^m, n \leq m$ be linearly independent vectors. The set*

$$L = \{a_1v_1 + ... + a_nv_n : a_1, ..., a_n \in \mathbb{Z}\}$$

is called the lattice L generated by $v_1, v_2, ..., v_n$.

Remark 1 *Any set of independent vectors that generate L is called a basis for L. Any two such bases have the same number of elements, which is called the dimension of L.*

Proposition 1 *Let $V = \{v_1, ..., v_n\}$ and $W = \{w_1, ..., w_n\} \in L$ two bases for L. We form the $n \times m$ matrices:*

$$V = \begin{pmatrix} -v_1- \\ \vdots \\ -v_n- \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} -w_1- \\ \vdots \\ -w_n- \end{pmatrix}$$

named bases matrices. Then there exist an $n \times n$ invertible matrix with integer entries U (which is equivalent to the fact that $det(L) = \pm 1$) such as $W = UV$.

The two fundamental hard problems in a lattice $L \subset \mathbb{R}^m$ used in cryptography are **The Shortest Vector Problem** (SVP) and **The Closest Vector Problem** (CVP). The first one's task is finding a shortest nonzero vector u in the lattice L (which minimizes the Euclidean norm $||u|| = \sqrt{u_1^2 + ... + u_m^2}$, where $u = (u_1, ..., u_m)$, while the second one's is finding a vector $v \in \mathbb{R}^m$ with $||w - v||$ being the smallest from all the values $||w - x||$ with $x \in L$, where $w \in \mathbb{R}^m \setminus L$ is a given non-lattice vector (or, equivalently, finding the vector $v \in L$ that is closest to the given vector $w \in \mathbb{R}^m \setminus L$).

In the NTRU cryptosystem and in learning with errors based settings, the computations are made in rings of the form $R = \mathbb{Z}[\mathbb{X}]/(f)$ where f is a polynomial with special properties such as the problems that the system relies on are mapped in a lattice in which the SVP and CVP problems are difficult.

2.2 NTRU Lattices

NTRU lattices are a particular case of cyclic lattices [Lyu08]. Their advantage is the fact that solving problems in quotient rings is reduced to solving versions of SVP (the NTRUEncrypt and NTRUMLS signature schemes were build upon these properties). They were introduced in [Mic].

Definition 2 ***Convolution Polynomial Rings****: Fix $N \in \mathbb{N}^*$. The quotient ring $R = \mathbb{Z}[x]/(x^N - 1)$ is named the ring of convolution polynomials of rank N. The ring $R_q = \mathbb{Z}_q[x]/(x^N - 1)$ is named similarly by the ring of convolution polynomials modulo q of rank N.*
Every element of R or R_q ($a(x)$) is represented in an unique form $a_0 + a_1 x + a_2 x^2 + ... + a_{N-1} x^{N-1}$ and we can associate it a vector from Z^N or Z_q^N, $(a_0, ..., a_{N-1})$.

Remark 2 *The vector corresponding to the sum of two polynomials $a(x)$ and $b(x)$ is the sum of their corresponding vectors.*
When it comes to multiplication, the corresponding resulted vector is given by the following formula: $c_k = \sum_{i+j \equiv k (mod\, N)} a_i b_{k-i}$

- In general, there are used polynomials with coefficients in the interval $\left(-\frac{q}{2}, \frac{q}{2}\right]$, and these are obtained by *center-lifting*, which is defined below:

Definition 3 *Let $a(x) \in R_q$. The center-lift of $a(x)$ to R is the unique polynomial $a'(x) \in R$ satisfying $a(x) mod\, q = a'(x)$, whose coefficients are chosen in the interval $\left(-\frac{q}{2}, \frac{q}{2}\right]$.*

The inverse of a polynomial in R_q is defined as follows:

Definition 4 *Let q be a prime. Then $a(x)$ has a* ***multiplicative inverse*** *if and only if $gcd(a(x), x^N - 1) = 1$ in $\mathbb{Z}_q[x]$, and it is computed using the Euclidean Algorithm to find polynomials $u(x), v(x) \in \mathbb{Z}_q[x]$ satisfying $a(x)u(x) + (x^N - 1)v(x) = 1$. Then $a(x)^{-1} = u(x)$.*

2.3 Ideal lattices

Ideal lattices [LPR10] are a generalisation of cyclic lattices, and they are characterised by their small number of parameters used in describing them.
They are lattice that correspond to ideals in rings of the form $\mathbb{Z}[X]/(f)$ for some irreducible polynomial f of degree n. They began as a generalisation of the cyclic lattices, and they are the basis of the Ring Learning With Errors problem.

2.4 Ring Learning-With-Errors problem

Another elegant approach, qualitative regarding performance is using the Ring Learning-With-Errors problem [LPR10], which may be reducible to the **SVP** problem in an ideal lattice.
The setup:

- It takes place in a ring of the form $R^{q^n} = \mathbb{Z}_q/(X^n + 1)$, where $n = 2^k$ and q is a prime.
- The polynomials we are interested in are those with small coefficients (the infinity norm is small). The coefficients used are from the set $\{-\frac{q-1}{2}, ..., 0, ..., \frac{q-1}{2}\}$, and choosing this kind of polynomials can be done by considering a much less integer than q, let's say, b. If we randomly choose coefficients from the set $\{-b, -b+1, ..., 0, ..., b-1, b\}$, we obtain a desired polynomial.

Let:

- $a_i(x) \in R^{q^n} = \mathbb{Z}_q/(X^n + 1)$ a set of random but **known** polynomials with coefficients from $\{-\frac{q-1}{2}, ..., 0, ..., \frac{q-1}{2}\}$.
- $e_i(x) \in R^{q^n} = \mathbb{Z}_q/(X^n + 1)$ a set of random but **unknown** polynomials with coefficients from $\{-\frac{b-1}{2}, ..., 0, ..., \frac{b-1}{2}\}$.
- $s(x)$ a small random **unknown** polynomial with coefficients from $\{-\frac{b-1}{2}, ..., 0, ..., \frac{b-1}{2}\}$.
- $b_i(x) = a_i(x) \cdot s(x) + e_i(x)$.

There are two versions of the RLWE problem:

- **The search version** means finding the **unknown polynomial** $s(x)$ given only the list of polynomial pairs $(a_i(x), b_i(x))$.

- **The decision version**: given a list $(a_i(x), b_i(x))$, the task is to determine whether $b_i(x)$ were constructed as $b_i(x) = a_i(x)s(x) + e_i(x)$ or were randomly generated from R^{q^n}.

Remark 3 *For increasing the difficulty of the problem we choose $n = 2^k$ (generally, if the ideal is generated by a cyclotomic polynomial ϕ, the problem is equivalent to to finding a short vector, but not necessarily the shortest vector, in an ideal lattice formed from elements of $\mathbb{Z}[X]/(\phi[X])$ represented as integer vectors, but we choose polynomials of this form for practical reasons) and q to be congruent to 1 modulo $2n$.*

Remark 4 *In a **RLWE** based public key algorithm the **private key** is the pair of small polynomials (with respect to the infinity norm) and the corresponding **public key** is the pair of polynomials $(a(x), b(x))$ (the list mentioned above contains a single value). Given $a(x), b(x)$, it should be computationally infeasible to recover the polynomial $s(x)$. Also, when choosing the polynomials $e_i(x)$, they can be multiplied by a constant $t \in \mathbb{Z}^+$ such as $gcd(q, t) = 1$, and the hardness assumption still holds.*

Remark 5 *In the following systems, the shared secret will be computed by multiplying one part's s with the other part's public key. Let's say we used the value $t = 2$. The two obtained values are close, so having to agree on the shared secret, one part sends a signal value that indicates whether its obtained value lies in $[-q/4, q/4]$ or not. Depending on this, the other participant computes the secret. But this strategy leaks an information about the key, so in there was used the following notion:*

Definition 5 *An algorithm $f()$ is a robust extractor on $\mathbb{Z}_q$ with error tolerance δ with respect to a function h if:*

- *f takes as input $x \in \mathbb{Z}_q$ and a signal $\alpha \in \{0, 1\}$ and outputs $k = f(x, \alpha) \in \{0, 1\}$.*
- *h takes as input $y \in \mathbb{Z}_q$ and a signal $\alpha = h(y) \in \{0, 1\}$*
- *$f(x, \alpha) = f(y, \alpha)$ for any $x, y \in \mathbb{Z}_q$ such as $x - y$ is even and $|x - y| \leq \delta$ where $\alpha = h(y)$.*

2.5 NTRU cryptosystem

We will use the following notation:

$$\tau(d_1, d_2) = \left\{ a(x) \in R : \begin{array}{l} a(x) \text{ has } d_1 \text{ coefficients equal to } 1 \\ a(x) \text{ has } d_2 \text{ coefficients equal to } -1 \\ a(x) \text{ has all other coefficients equal to } 0 \end{array} \right.$$

Polynomials in $\tau(d_1, d_2)$ are called *ternary polynomials.*

- Alice or some trusted authority chooses public parameters (N, p, q, d) with the following properties:

1. N is prime.
2. $gcd(N, q) = gcd(p, q) = 1$.
3. $q > (6d + 1)p$.

- She chooses the **private key**: two randomly chosen polynomials $f(x) \in \tau(d + 1, d)$ and $g(x) \in \tau(d, d)$.
- She computes the **public key** as follows:
  1. Computes the inverses $\mathbf{F}_q(x) = f(x)^{-1}$ in R_q and $\mathbf{F}_p(x) = f(x)^{-1}$ in R_p. If f is not invertible in one of the rings R_p or R_q, she chooses another.
  2. She computes $h(x) = \mathbf{F}_q(x)g(x)$ in R_q, which is her **public key**.
- Bob wants to send Alice a message so he transforms it into a polynomial from R_p (or equivalently, an N-dimensional vector with entries from Z_p), and then he *center-lifts* it.
- He chooses a random polynomial $r \in \tau(d, d)$ and computes $e(x) \equiv ph(x)r(x) + m(x)\ (mod\, q)$, so $e(q)$ is in the ring R_q, and this is Bob's **ciphertext**.
- Once receiving $e(x)$ from Bob, Alice firstly computes $a(x) \equiv f(x)e(x)(\text{mod q})$.
- Then she *center-lifts* it and computes $b(x) \equiv \mathbf{F}_p(x)e(x)(\text{mod p})$, which is equal to the plaintext $m(x)$.

3 Approaches based on modifying the ntru protocol

There exist two approaches of this kind: the one presented in [GK15] and the one from [SWZ16] . In each one of them it is described a slight modification of the ntru protocol such as the exchanged secret values are additionally encapsulated using either NTRU or Ring Learning With Errors keys. We notice that they choose hybridization of the original protocol instead of replacing the actual values so as the current cryptographic infrastructure to not be affected.

3.1 HybrydOR

The protocol's setup is the following:

- H_{st}, H_1, H_2 and a PRF, where H_{st} is a collision-resistant hash function, H_1 is a randomness extractor, H_2 is a random oracle, and PRF is a pseudorandom function for the key confirmation message.
- f^R, a robust extractor and h^R, a randomized algorithm used for the signal value of the robust extractor.

The protocol takes place after the following pattern:

- The system parameters are the cyclic group G along with its order p and its generator q (the group of points in Curve25519) and the public parameters for LWE: the exponent n, the prime number q, $t = 2$, the description of the ring R.
- The server chooses his static keys: $a \in R_q$, (from LWE), which is a public key, and the key pair $(b, B = g^B)$.
- The client chooses its ephemeral parameters: the key pair $x, X = g^x$, s_X, e_X and the corresponding public key $p_X = ar_X + te_X$. It sends a message containing $X|p_A$.
- The server computes an ephemeral key pair r_Y, e_Y, $p_Y = ar_Y + te_Y$, and e'_Y, with the purpose of calculating the key: $k_Y = p_A r_Y + te'_Y$, $\alpha = h^R(k_Y)$, and the key has the components $k_1 = f^R(k_Y, \alpha)$, $k_2 = g^{xb}$. Computes $(sk_m, sk) = H_1(k_1, p_X, p_Y, ID) \oplus H_2(k_2, X, Y, ID)$, $tag = PRF(sk_m, ID, p_Y, \alpha, p_X, X, "server")$ and sends $p_X, \alpha, tag, HybrOR$ to the client.
- The client computes $k_X = p_Y r_X + te_X$, $k_1 = f^R(k_X, \alpha)$, $k_2 = g^{xb}$, $(sk_m, sk) = H_1(k_1, p_X, p_Y, ID) \oplus H_2(k_2, X, Y, ID)$, $tag = PRF(sk_m, ID, p_Y, \alpha, p_X, X, "server")$.

3.2 NtruTOR

The proposal follows the same pattern, i.e. using both Diffie-Hellman and NTRU keys:

- The public parameters are the existent ones and the NTRU-specific: N, p, q, d.
- The server generates its static key: $b, B = g^b$. Its identity is ID.
- The client generates ephemeral key pairs $x, X = g^x$, $(f_X, g_X), h_X$ and sends a message containing them.
- The server generates the ephemeral keys $y, Y = g^y$ and computes $s_0 = h(X^b)$, $s_1 = X^y$, and generates a shared key k which is picked randomly and encrypted using NTRU with the public key h_X, obtaining e. Computes $T = ID|B|X|Y|h_X|e$, and computes a pseudorandom key (prk) using $s_0|s_1|k$ and T. Sends the authentication tag $PRF(prk, t_{auth})$ (t_{auth} being as in the TOR specifications) along with the other ntor specific information.
- The client computes $s_0 = H(B^x)$, $s_1 = Y^x$ and decrypts k, checking the remaining values.

The two protocols rely on the Gap Diffie-Hellman assumption of the cyclic group and on the strenght of the lattice-based primitives. The difference in computation speed from the original ntor is not a drawback, but it is recommended to decrease the length of the prime numbers used, as it can reduce the communication overhead of the protocol.

3.3 Post Quantum Key Encapsulation approach

In [Tuj+20], is proposed a study in which post quantum key-related algorithms are used and compared, being evaluated six algorithms that ensure level 1 NIST security. Notable is the fact that there is no hybridization of the protocol, the infrastructures remaining the same. The migration to quantum-safe TOR starts with the update of cryptographic algorithms that involve long-term and medium-term keys such as the identity key. Such a migration naturally results in additional cost in terms of CPU and bandwidth, being analysed the cost of key generation, key encapsulation and decapsulation and comparing them to the current RSA and ECC primitives.
The compromise of the used keys leads to various types of attacks: harvest-and-decrypt, authentication of connections that should be not allowed, creation of new signing keys, impersonations, poisoning the Descriptors so as all the traffic to be redirected to corrupt routers etc. The key encapsulation schemes evaluated in the ntor context were RSA-1024, RSA-2048, Kyber512, NewHope-512-CCA, NTRU-HPS-2048-509, Sike-p503, Frodo-640-AES, Frodo-640-SHAKE. As a result, the keys generated by the post-quantum algorithms are larger, so they have an effect on the network load, because they are sent to the Directory Authority that forwads them to the clients. Also, larger ciphertexts increase the network load, having a negative effect on the stability of the network: Frodo-640-AES, Frodo-640-SHAKE have ciphertexts of length of 9720 bytes.
The lattice-based schemes require less CPU cycles for key generation than RSA does, and SIKE is more efficient than RSA 2048.
Also, key generation affects the nodes and the client regarding response time. Key encapsulation and decapsulation impact the communication time between the node and the client because of the time needed to encapsulate/decapsulate messages. NTRU requires more CPU cycles than RSA for encapsulation but less CPU cycles for decapsulation. Isogeny-based SIKE requires more cycles for both operations, being 48 times more costly than RSA-2048. In average, lattice based schemes are the best choice, managing to balance the time consumed for all three operations. Regarding the ciphertext overhead, SIKE is the best choice as the ciphertext size is exactly the size of one TOR cell: 512 bytes.
As future work, there is proposed the code-based BIKE [Ara+17] scheme and testing the remaining isogeny-based and lattice-based schemes.

4 Conclusion

In this paper are analysed the current cryptographic approaches for making TOR quantum-resistant. This aspect pose a challenge, as the present quantum schemes can be computationally heavy, they can add a load to the network and are difficult to add to or replace the current standards. A solution is hybridization using Ring Learning With Errors approaches as the addition of the afferent new parameters doesn't increase computations, moreover, it can increase the efficiency of the computations made by the nodes. Another approach is to generate keys using post quantum algorithms, where the best fit is made by the lattice based systems. As a future work, key generation and encryption are to be tried using code-based approaches.

References

[SDM04] Paul Syverson, Roger Dingledine, and Nick Mathewson. "Tor: The second generation onion router". In: *Usenix Security.* 2004, pp. 303–320.

[Lyu08] Vadim Lyubashevsky. "Lattice-Based Identification Schemes Secure Under Active Attacks". In: *Public Key Cryptography - PKC 2008.* Vol. 4939. Lecture Notes in Computer Science. Springer. 2008, pp. 162–179.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer. 2010, pp. 1–23.

[GSU13] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. "Anonymity and one-way authentication in key exchange protocols". In: *Designs, Codes and Cryptography* 67.2 (2013), pp. 245–269.

[GK15] Satrajit Ghosh and Aniket Kate. "Post-quantum forward-secure onion routing". In: *International Conference on Applied Cryptography and Network Security.* Springer. 2015, pp. 263–286.

[SWZ16] John M Schanck, William Whyte, and Zhenfei Zhang. "Circuit-extension handshakes for Tor achieving forward secrecy in a quantum world". In: *Proceedings on Privacy Enhancing Technologies* 2016.4 (2016), pp. 219–236.

[Ara+17] Nicolas Aragon et al. "BIKE: bit flipping key encapsulation". In: (2017).

[Tuj+20] Zsolt Tujner et al. *QSOR: Quantum-Safe Onion Routing.* 2020. arXiv: `2001.03418 [cs.CR]`.

[Mic] Daniele Micciancio. "Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions from Worst-Case Complexity Assumptions". In: *FOCS 2002.*

[NTS] NTSpec. *ntor.* `https://gitweb.torproject.org/torspec.git/tree/proposals/216-ntor-handshake.txt`. Accessed: 2010-09-30.