Post-Quantum Secure Onion Routing
(Future Anonymity in Today’s Budget)

Satrajit Ghosh† Aniket Kate‡
†Indian Statistical Institute (ISI), Kolkata, India ‡CISPA, Saarland University, Germany

Abstract

The onion routing (OR) network Tor provides anonymity to its users by routing their encrypted traffic through three proxies (or nodes). The key cryptographic challenge, here, is to establish symmetric session keys using a secure key exchange between the anonymous users and the selected nodes. The Tor network currently employs a one-way authenticated key exchange (1W-AKE) protocol (ntor) for this purpose. Nevertheless, (ntor) as well as other known 1W-AKE protocols rely solely on some classical Diffie-Hellman (DH) type assumptions for their (forward) security, and thus privacy of today’s anonymous communication cannot be ensured once quantum computers arrive.

In this paper, we demonstrate utility of quantum-secure lattice-based cryptography towards solving this problem for onion routing. In particular, we present a novel hybrid 1W-AKE protocol (HybridOR) that is secure under the lattice-based ring learning with error (ring-LWE) assumption as well as the gap DH assumption. Due to its hybrid design, HybridOR is not only resilient against quantum attacks but also allows the OR nodes to use the current DH public keys and subsequently requires no modification to the the current Tor public key infrastructure. Moreover, thanks to the recent progress in lattice-based cryptography in the form of efficient ring-based constructions, our protocol is also computationally more efficient than the currently employed 1W-AKE protocol (ntor), and it only introduces small and manageable communication overhead to the Tor protocol.

1 Introduction

Lattice-based cryptographic constructions have drawn an overwhelming amount of research attention in the last decade [7,33,35,38,42]. Their strong provable worst case security guarantee, apparent resistance to quantum attacks, high asymptotic efficiency and flexibility towards realizing powerful primitives such as fully homomorphic encryption [23] have been the vital reasons behind the popularity. Although the powerful primitives (such as fully homomorphic encryption) are still very far from being ideal for practical use, several recent efforts have demonstrated that performance of lattice-based constructions for basic encryption and authentication primitives is comparable with (and sometimes even better than) performance of corresponding primitives in the classical RSA or DLog settings [1,28,31,33]. As a result, some work has now started to appear towards developing lattice-based version of real-world cryptographic protocols such as TLS/SSL [6]. In this work, we explore the utility of quantum-secure yet highly efficient lattice-based cryptography to anonymous communication networks (ACNs).

Over the last three decades, several ACNs have been proposed and few implemented [10,13,17,20,25,40,43,44]. Among these, with its more than two million users and six thousand onion routing (OR) proxies spread all across the world, the OR network Tor [17,47] has turned out to be a huge success. Today, along with anonymous web browsing and hosting, Tor is also extensively used for censorship-resistant communication [15].

A typical realization of an OR network (such as Tor) consists of an overlay network of proxies (or nodes) that routes their users’ traffic to their Internet-based destinations. A user chooses an ordered sequence of OR nodes (i.e., a path) through the OR network using a path selection strategy, and constructs a cryptographic circuit using a public-key infrastructure (PKI) such that every node in the path shares a symmetric session key with the user. While employing the circuit to send a message anonymously to a destination, the user forms an onion by wrapping the message in multiple layers of symmetric encryption such that upon receiving the onion every node can decrypt (or remove) one of the layers and then forward it to the next node in the circuit.

From the cryptographic point of view, the key challenge with an OR protocol is to securely agree upon the required session keys so a user can individually authenticate the nodes in her circuits while maintaining her anonymity (except from the first node). Since its inception, Tor employed an interactive forward-secret key-exchange proto-
col called the Tor authentication protocol (TAP) to agree upon those session keys in a telescoping (or multi-pass) construction [17]. Due to its atypical use of CPA-secure RSA encryption, TAP was considered weaker in terms of performance as well as security [24]. Recently, Goldberg, Stebila and Ustaoglu [26] formalized the OR key agreement security by introducing the concept of one-way authenticated key exchange (1W-AKE), and designed a provably secure 1W-AKE protocol called ntor. With its significantly better computation and communication efficiency, ntor has since replaced TAP in the real-world Tor implementation [16].

Nevertheless, security of ntor and other 1W-AKE protocols [3, 9, 29] requires some variant of Diffie–Hellman (DH) assumption in the classical discrete logarithm (DLog) setting. As the DLog problem and all of its weaker DH variants can be solved in polynomial time (in the security parameter) using quantum computers, the security of these 1W-AKE constructions and subsequently the confidentiality and anonymity of the OR communications will be broken in the post-quantum world. Note that the current 1W-AKE protocols are also not forward-secure against the quantum attacks; the confidentiality and anonymity of even today’s OR communications can be violated once quantum computers arrive.

Although this raises concern regarding the privacy of today’s anonymous communication in the future, making drastic modifications to the current OR infrastructure by replacing the current 1W-AKE construction with a lattice-based construction may be injudicious; e.g., in Tor, this will require completely changing the public key infrastructure (PKI). As a result, it presents an interesting challenge to define a lattice-based 1W-AKE protocol that offers security in the post-quantum world without significantly affecting the current cryptographic infrastructure and performance.

1.1 Contribution

In this paper, we resolve this challenge by presenting a novel hybrid 1W-AKE protocol (HybridOR) that combines lattice-based key exchange with the standard DH key exchange. The channel security of HybridOR relies on the (standard) ring variant of learning with error (ring-LWE) assumption and the gap Diffie–Hellman (GDH) assumption, while its forward secrecy and the security against an man-in-the-middle impersonator rely respectively on the ring-LWE assumption and the GDH assumption. Interestingly, while achieving this enhanced security properties, HybridOR does not require any modifications to the current Tor public keys or the directory infrastructure. We observe that HybridOR is computationally more efficient than the currently employed ntor protocol; in particular, the efficiency improvement on both the client and the node sides is nearly 33%. Although this improved security and efficiency comes at the cost of increased communication, both the client and the node will have to communicate three Tor cells each, which we find to be manageable for the Tor network today.

Finally, along with apparent resistance to quantum attacks and the worst case security guarantee, as our HybridOR protocol is a 1W-AKE, it can also be used to realize a universally composable (UC) OR protocol [2].

Outline. We start our discussion by presenting a brief overview of the OR protocol, the gap Diffie–Hellman problem, and the learning with errors problem in Section 2 and describe the correctness, security and anonymity requirements of a 1W-AKE protocol in Section 3. We then present our HybridOR protocol in Section 4 and shows that HybridOR indeed constitutes a 1W-AKE protocol in Section 5. Section 6 compares the computational efficiency and the message sizes of the HybridOR protocol with the ntor protocol. Finally, Section 7 concludes and discusses some future work.

2 Background

In this section, we present a brief overview of the OR protocol employed by Tor, the GDH assumption in the DLog setting, and describe the learning with errors problem in the lattice-based setting.

2.1 The Onion Routing Protocol

In the original OR protocol [27, 40, 46] circuits were constructed in a non-interactive manner. In particular, a user created an onion where each layer contained a symmetric session key for an OR node and the IP address of the successor OR node in the circuit, all encrypted with the original node’s public key such that each node can decrypt a layer, determine the symmetric session key and forward the rest of the onion along to the next OR node. Unless public keys are rotated frequently, this approach cannot guarantee forward security for the anonymous communication; thus, in the second generation OR network [17] (i.e., Tor), circuits are constructed incrementally and interactively, where symmetric session keys are established using a forward-secure Diffie–Hellman (DH) key exchange involving the OR node’s public key. In the second generation Tor protocol, circuits are constructed using the Tor authentication protocol (TAP) involving a CPA-secure RSA encryption and a DH key exchange. Currently, the third generation Tor network employs the

\footnote{TAP has been proven secure only in a weaker security model [24].}
provable secure against the gap Diffie-Hellman (GDH) assumption [36] and significantly more efficient ntor protocol [26].

In related efforts, Backes et al. [2] observe that, with minor modifications, universally composable (UC) security [8] is possible for the existing Tor protocol, if the employed key agreement protocol is a (one-way anonymous) one-way authenticated key exchange [26].

One-way Authenticated Key Exchange—1W-AKE.

Goldberg, Stebila and Ustaoglu introduce a security definition of one-way authenticated key exchanges (1W-AKE) to facilitate design of provably secure session key agreement protocols for onion routing [26]. They also fixed a key agreement protocol proposed in [37] to obtain a provably secure construction called the ntor protocol, which has replaced the TAP protocol in the current Tor network.

In ntor protocol, as described in Figure 1, the client sends a fresh ephemeral key \( g^x \) to the node. The node computes and sends a fresh ephemeral key \( g^y \) to the client and calculates the session key as \( H((g^x)^y, (g^x)^b) \), where \( b \) is the long term secret key of the node.

Recently, Backes, Kate, and Mohammadi [3] introduced a 1W-AKE protocol Ace that improves upon the computational efficiency of ntor. In Ace (see Figure 1) the client sends two fresh ephemeral keys \( g^{x_1} \) and \( g^{x_2} \) to the node. The node sends one fresh ephemeral key \( g^b \) to the client. The client and node compute the shared secret as \( g^{x_1 b + x_2 y} = (g^b)^{x_1} (g^y)^{x_2} = (g^{x_1})^b (g^{x_2})^y \). The source of efficiency in Ace comes from the fact that one can do two exponentiations at the same time using the multi-exponeniation trick.

In contrast to the above interactive 1W-AKE protocol, a single-pass construction using a non-interactive key exchange is possible as well. However, achieving forward secrecy of the user’s circuits without regularly rotating the PKI keys for all Tor nodes is not possible [29], and the periodic public key rotation should be avoided for scalability reasons. There have been attempts to solve this problem by leveraging the identity-based setting [29] or the certificate-less cryptography setting [9]. Nevertheless, as discussed in [2], the key authorities required in these constructions can be difficult to implement in practice.

2.2 Gap Diffie-Hellman Problem

Let \( G \) be a group with large prime order \( p \) and \( g \in G \) be the generator of the group. Given a triple \((g, g^a, g^b)\) for \( a, b \in Z_p \), the gap version of Diffie-Hellman (GDH) problem is to find the element \( g^{ab} \) with the help of a Decision Diffie-Hellman (DDH) oracle [36]. The DDH oracle \( O_{\text{ddh}} \) takes input as \((G, g, g^a, g^b, z)\) for some \( z \in G \) and tells whether \( z = g^{ab} \) or not, that is whether the tuple is a DH tuple or not. Solving GDH problem in \( G \) is assumed to be hard problem. More formally,

**Definition 1** (GDH problem). For all algorithm \( A \), the advantage of solving GDH in the group \( G \) is defined as,

\[
\text{Adv}_{A}^{\text{gdh}} = \Pr[A^{O_{\text{ddh}}}(p, g, g^a, g^b) = g^{ab}, (a, b) \in \mathbb{Z}_p^2].
\]

The GDH assumption states that \( \text{Adv}_{A}^{\text{gdh}} \) is a negligible function of the security parameter for all PPT algorithms \( A \).

2.3 Learning With Error Problem

Learning with error (LWE) is the problem of distinguishing noisy random linear equations from truly random ones, for a small amount of noise. This is a hard machine learning problem over lattices [42]. The LWE problem has been shown to be as hard as some worst case lattice problems [42]. Since then, different variants of the LWE problem have been employed to design lattice-based cryptosystems [32, 34, 41, 42]. The main drawback of schemes based on LWE is that they are based on matrix operations, which are quite inefficient and result in large key sizes. To overcome these problems, special lattices with extra algebraic structures are used to construct cryptographic applications.

**Ring-LWE problem.** Lyubashevsky [33] propose an algebraic variant of LWE, ring-LWE, to reduce computation, communication and storage complexity. The Ring-LWE problem is defined over a polynomial ring \( \mathbb{Z}_q \) with coefficient in \( \mathbb{Z} \). Let \( \mathbb{Z}[x] \) denote the set of polynomials with integer coefficients in \( \mathbb{Z} \). Consider \( f(x) = x^n + 1 \in \mathbb{Z}[x] \), where the degree of the polynomial \( n \geq 1 \) is a power of 2, making \( f(x) \) irreducible over the \( \mathbb{Z} \). Let \( \mathbb{R} = \mathbb{Z}[x]/\langle f(x) \rangle \) be the ring of integer polynomials modulo \( f(x) \). Elements of \( \mathbb{R} \) can be represented by integer polynomials of degree less than \( n \). Let \( q \equiv 1 \mod 2n \) be a sufficiently large public prime modulus (bounded by a polynomial in \( n \)), and let \( \mathbb{R}_q = \mathbb{Z}_q[x]/\langle f(x) \rangle \) be the ring of integer polynomials modulo both \( f(x) \) and \( q \). The ring \( \mathbb{R}_q \) contains all the polynomials of degree less than \( n \) with coefficient in \( \mathbb{Z}_q \), along with two operations, polynomial addition and multiplication modulo \( f(x) \).

\( \chi \) denotes the error distribution over \( \mathbb{R} \), which is concentrated on small elements of \( \mathbb{R} \). Details of the error distribution for the security and the correctness of the system can be found in [33]. We denote \( \mathbb{D}_{\chi} \) as the ring-LWE distribution over \( \mathbb{R}_q \times \mathbb{R}_q \), obtained by choosing uniformly
The ntor Protocol

\[
x \leftarrow_R \mathbb{Z}_p \quad \xrightarrow{g^x} \quad y \leftarrow_R \mathbb{Z}_p
\]

\[H((g^y)^x, (g^b)^x) = H(g^{ux}z, g^{ux})\]

\[\text{(established session key } H(g^{ux}, g^{ub})\text{)}\]

The Ace Protocol

\[
x, x_1, x_2 \leftarrow_R \mathbb{Z}_p \quad \xrightarrow{g^{x_1}, g^{x_2}} \quad y \leftarrow_R \mathbb{Z}_p
\]

\[H((g^{x_1})^{x_2}, (g^{s_1})^{x_2}) = g^{x_1+s_2}\]

\[\text{(established session key } H(g^{x_1+s_2})\text{)}\]

Figure 1: A comparative overview of the current 1W-AKE protocols. For the sake of readability, we neglected the session information used for the key derivation and the key confirmation message.

random \(a \leftarrow \mathbb{R}_q, e \leftarrow \chi\) and output \((a, a \cdot s + e)\), for some \(s \leftarrow \mathbb{R}_q\).

Search ring-LWE problem. The search version of ring-LWE is, for uniformly random \(s \leftarrow \mathbb{R}_q\), given a \(\text{poly}(n)\) number of independent samples from \(\mathbb{D}_{s, \chi} \subseteq \mathbb{R}_q \times \mathbb{R}_q\), to find \(s\).

Decision ring-LWE problem. The decision version of ring-LWE is to distinguish between two distributions, \(\mathbb{D}_{s, \chi}\), for uniformly random \(s \leftarrow \mathbb{R}_q\) and a uniformly random distribution in \(\mathbb{R}_q \times \mathbb{R}_q\) (denoted by \(U_{\mathbb{R}_q \times \mathbb{R}_q}\)), given a \(\text{poly}(n)\) number of independent samples. More formally,

Definition 2 (Decision ring-LWE problem). The decision ring-LWE problem for \(n, q, \chi\) is to distinguish the output of \(\mathbb{O}_{s, \chi}\) oracle from the output of an oracle that returns uniform random samples from \(\mathbb{R}_q \times \mathbb{R}_q\). If \(A\) is an algorithm, the advantage of \(A\) is defined as

\[\text{Adv}_A^{\text{drake}} = |\Pr[A^{\mathbb{O}_{s, \chi}}(\cdot)] - \Pr[A^{U_{\mathbb{R}_q \times \mathbb{R}_q}}(\cdot)]|\]

The decision ring-LWE assumption states that for every PPT adversary \(A\), \(\text{Adv}_A^{\text{drake}}\) is negligible.

The hardness results for the LWE problem are described in [33][38][42]. Brakerski et al. [7] show the classical hardness of the LWE problem. Ding et al. [14] mention that for any \(t \in \mathbb{Z}^+\), such that \(\gcd(t, q) = 1\), the LWE assumption still holds if we choose \(b = (a, s) + te\). We use \(t = 2\) for our construction.

It is important to note that ring-LWE samples are pseudorandom even when the secret \(s\) is chosen from the error distribution [1][35]. Ducas and Durmus [18] show that the ring-LWE problem is hard in any ring \(\mathbb{Z}[x]/(\Phi_m)\), for any cyclotomic polynomial \(\Phi_m(x)\).

Robust extractors. One of the important problems with the lattice-based key exchange protocols is the error correction of the shared secret. There are different methods [14][22] to agree on a shared secret from noisy shared secret values. For our construction we adopt the method due to Ding et al. [14] and recall the corresponding concept of robust extractors and the signal functions below.

Intuition. Let Alice sends \(p_a = a_s + 2e_a\) to Bob, and Bob sends \(p_B = a_s + 2e_B\) to Alice, where \(e_A, e_B \in \chi\) and \(a, s_A, s_B \in \mathbb{R}_q\). Here, \(s_A\) and \(s_B\) are the secret keys of Alice and Bob respectively. To compute the shared secret Alice computes \(K_A = p BS_A + 2e_A s_A\) mod \(q\) and Bob computes \(K_B = p AS_B + 2e_B s_B\) mod \(q\). Clearly \(K_A - K_B\) is even and small. Using the robust extractor as explained in [14], it is possible for Alice and Bob to agree on the same value once they have \(K_A\) and \(K_B\) respectively. To achieve this goal Bob has to send a signal value indicating whether \(K_B\) lies in \([-q/4, q/4]\) \(\cap \mathbb{Z}\) or not. If \(K_B\) lies in \([-q/4, q/4]\) \(\cap \mathbb{Z}\) we can write \(K_B = K_A + 2(e_A S_B - e_B S_A)\) mod \(q\). Now \(2(e_A S_B - e_B S_A) \leq q/4\), then:

\[K_B = K_A + 2(e_A S_B - e_B S_A)\] mod \(q\),
\[K_B = K_A\] mod \(q\) + \(2(e_A S_B - e_B S_A)\),
\[(K_B \mod q) \mod 2 = (K_B \mod q) \mod 2\].

This is also true when \(K_B\) lies outside the interval \([-q/4, q/4]\) \(\cap \mathbb{Z}\). However, this type of deterministic extractor leaks the information whether \(K_B\) lies inside or outside a certain interval. To solve this problem Ding et. al. [14] propose a randomized signal generation.
algorithm that removes the bias of the distribution of the extracted key.

**Definition 3 (Robust Extractors).** An algorithm \( f(\cdot) \) is a robust extractor on \( \mathbb{Z}_q \) with error tolerance \( \delta \) with respect to a hint function \( h(\cdot) \) if:

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

We use the robust extractor as described in [14]. For \( q > 2 \) define \( \alpha_0 : \mathbb{Z}_q \rightarrow \{0, 1\} \) and \( \alpha_1 : \mathbb{Z}_q \rightarrow \{0, 1\} \) as follows:

\[
\alpha_0(x) = \begin{cases} 
0, & \text{if } x \in [-\frac{q}{4}, \frac{q}{4}] \setminus \{0\}; \\
1, & \text{otherwise.}
\end{cases}
\]

\[
\alpha_1(x) = \begin{cases} 
0, & \text{if } x \in [-\frac{q}{4} + 1, \frac{q}{4}] \cup \{0\}; \\
1, & \text{otherwise.}
\end{cases}
\]

The hint algorithm \( h(\cdot) \) generates the signal \( \alpha \) for some \( y \in \mathbb{Z}_q \) by tossing a random coin \( b \leftarrow \{0, 1\} \) and computing \( \alpha = h(y) = \alpha_b(y) \). Finally the robust extractor computes the common value as:

\[
f(x, \alpha) = (x + \alpha \cdot \frac{q - 1}{2} \mod q) \mod 2,
\]

where \( x \in \mathbb{Z}_q, |x - y| \leq \delta \) and \( x - y \) is even. In [14], the authors prove that \( f(\cdot) \) is a randomness extractor with respect to \( h(\cdot) \) for an odd integer \( q > 8 \) with error tolerance \( \delta = \frac{q}{4} - 2 \). Also if \( x \) is uniformly random in \( \mathbb{Z}_q \), then \( f(x, \alpha) \) is uniform in \( \{0, 1\} \), where \( \alpha = h(x) \).

It is easy to extend this notion for ring settings. Any element in \( \mathbb{R}_q \) can be represented by a degree \( n - 1 \) polynomial. For example any \( a \in \mathbb{R}_q \) can be written in the form \( a_0 + a_1x + \cdots + a_{n-1}x^{n-1} \). Then we can extend \( \alpha_0^R(a), \alpha_1^R(a) : \mathbb{R}_q \rightarrow \mathbb{R}_2 \) as follows:

\[
\alpha_0^R(a) = \sum_{i=0}^{n-1} a_0(a_0) x^i; \quad \alpha_1^R(a) = \sum_{i=0}^{n-1} a_1(a_0) x^i.
\]

The algorithm \( h^R(\cdot) \) can be defined in the same manner as \( h^R(a) = \alpha_0^R(a) \), for \( b \leftarrow \{0, 1\} \). Similarly define the extractor in the ring settings \( f^R(a, \alpha) : \mathbb{R}_q \rightarrow \mathbb{R}_2 \) as:

\[
f^R(a, \alpha) = (a + \alpha \cdot \frac{q - 1}{2} \mod q) \mod 2.
\]

**Authenticated key exchange in the lattice setting.** Fujioka et al. [21] provide a generic construction of authenticated key exchange (AKE) from a key-encapsulation mechanism (KEM). Their construction gives the first \( CK^+ \) secure AKE protocol based on ring-LWE problem, in a standard model. However due to the huge communication cost (\( \approx 139625 \) bytes) their lattice-based AKE is not suitable for real-world applications. In [22], Fujioka et al. propose a generic construction for AKE from OW-CCA KEMs in random oracle model. When instantiated with ring-LWE settings, their AKE protocol gives a much more efficient solution to the problem. Still, communication cost for [22] reaches about 10075 bytes. Peikert [39] proposes a new low-bandwidth error correction technique for ring-LWE based key exchange, and provides practical lattice-based protocols for key transport and AKE. Ding et al. [14] propose another method for error correction and design a passively secure DH-like key exchange scheme based on both the LWE and the ring-LWE problem. Zhang et al. [49] extend the above AKE protocol to ideal lattice settings, and their lattice-based AKE protocol gives weak perfect forward secrecy in the Bellare-Rogaway model [4]. Recently Bos et al. [6] demonstrate the practicality of using ring-LWE based key exchange protocols in real life systems. They employ lattice-based key exchange in TLS protocol. Their implementation reveals that the performance price for switching from pre-quantum-safe to post-quantum-safe key exchange is not too high and can already be considered practical, which further motivates our efforts towards defining a 1W-AKE protocol in the lattice setting.

### 3 1W-AKE Security Definition

Goldberg, Stebila and Ustaoglu [26] define the security requirements for a one-way authenticated key exchange (1W-AKE) protocol, which are refined in [3]. In this section we recall the security requirements for a 1W-AKE protocol between an anonymous client and an authenticated node.

Informally, a secure 1W-AKE protocol should respect the following properties:

**1W-AKE security.** An attacker should not be able to impersonate a node. In other words, an attacker cannot learn anything about the session key of an uncompromised session, even if it compromises several other sessions and introduces fake identities.

**1W-anonymity.** A node should not differentiate while communicating with two different clients.

A 1W-AKE protocol is a tuple of ppt algorithms \( \mathcal{AE} = (\text{SetUp}, \text{Init}, \text{Resp}, \text{CompKey}) \), where:

- **SetUp:** Generates the system parameters and the static long-term keys for the node.
- **Init:** The client calls Init to initiate the 1W-AKE protocol.
generates a fresh output message and stores already chosen keys. If the client computes public keys for the session (m, cs); send (m, cs) and ake_st^P(\Psi) = ⊥: (m', (k, *, st), \Psi)) \rightarrow Resp(sk_P, m, cs); result_st^P(\Psi) \leftarrow (k, st, *); send m'

upon send^P(\Psi, m) and ake_st^P(\Psi) ≠ ⊥: (N, st) \leftarrow ake_st^P(\Psi); check for a valid pk_N

\[(k, N, st) \leftarrow CompKey(pk_N, m, \Psi, (N, st))\]

erase ake_st^P(\Psi); result_st^P(\Psi) \leftarrow (k, N, st)

upon reveal_next^P: (x, X) \leftarrow Gen(1^\lambda); append (x, X) to cs; send X

upon partner^P(X): if a key pair (x, X) is in the memory then send x

upon sk_reveal^P(\Psi): if result_st^P(\Psi) = (k, N, st) then send k

upon establish_certificate(N, pk_N): register the public key pk_N for the party N

upon test^P(\Psi): (one time query)

\[(k, N, st) \leftarrow result_st^P(\Psi)\]

if k ≠ ⊥ and N ≠ * and \Psi is 1W-AKE fresh then if b = 1 then send k else send k' \leftarrow_R \{0, 1\}^{|k|}

Figure 2: 1W-AKE Security Challenger: Ch^KE_b(1^\lambda), where \lambda is the security parameter. If any invocation outputs ⊥, the challenger erases all session-specific information for that session and aborts that session.

- Resp: The node uses Resp to respond to an Init.
- CompKey: The client uses CompKey to verify the key-confirmation message and compute the key.

We assume that a PKI is given, i.e. for every party \(P_i \in \{P_1, \cdots, P_m\}\), all the other parties can obtain a (certified) public key \(pk_P\) such that only \(P_i\) itself also knows the corresponding secret key \(sk_P\).

### 3.1 Correctness

In a 1W-AKE protocol an anonymous client (denoted by \(\oplus\)) tries to establish a shared secret key with a party \(N\). The client calls \(\text{Init}(N, pk_N, cs)\), which returns an output message \(m\), session id \(\Psi\) and session state \(st\). The client sends \(m\) to \(N\). Init takes a queue \(cs\) as input, where \(cs\) stores already chosen keys. If \(cs\) is empty then \(Init\) generates a fresh output message \(m\). In response, \(N\) runs \(\text{Resp}(sk_N, N, m, cs)\) and outputs \((m', (k, \oplus, \vec{v}), \Psi_N)\), where \(m'\) is the response message to the client, \(k\) is the session key computed by \(N\), and \(\vec{v}\) contains ephemeral public keys for the session \(\Psi_N\). On receiving \(m'\), the client computes \((k', N, \vec{v}')\) by calling \(\text{CompKey}(pk_N, m', \Psi, st)\), where \(k'\) is the session key computed by the client and \(\vec{v}'\) is the list of ephemeral public keys. As described in [3], an AKE protocol is a correct 1W-AKE protocol if the following condition holds for every party \(N\):

\[
\Pr[(m, st, \Psi) \leftarrow \text{Init}(N, pk_N, cs), (m', (k, \oplus, \vec{v}), \Psi_N) \leftarrow \text{Resp}(sk_N, N, m, cs), (k', N, \vec{v}') \leftarrow \text{CompKey}(pk_N, m', \Psi, st) : k = k' \land \vec{v} = \vec{v}'] = 1.
\]

### 3.2 1W-AKE Security

The goal of the adversary in the 1W-AKE security experiment is to distinguish the session key of an uncompromised session from a random key. In other words, it requires an active attacker to not learn anything about the key and to not able to impersonate an honest node.

In the security game, the challenger \(Ch^KE\) represents honest parties \((P_1, \cdots, P_n)\) and allows the attacker a fixed set of queries described in Figure 2. The challenger internally runs the 1W-AKE algorithm, and simulates each party. For the challenge, the adversary asks \(Ch^KE\) for the session key of an uncompromised session \(\Psi\) for a party \(P\) by querying \(\text{test}^P(\Psi)\). \(Ch^KE\) tosses a random coin and depending on the output of the toss sends the correct session key or a randomly chosen session key to the attacker. The attacker’s task is to determine whether that corresponds to the session \(\Psi\) or is random.

For triggering the initiation session, triggering the response to a key exchange, and for completing a key exchange, challenger allows the adversary to query \(send^P(\cdot, m)\). For the compromising parties, the attacker can query using three different type of messages:

- reveal_next^P: The attacker can ask the party \(P\) to reveal the next public key that will be chosen.
- partner^P(X): The attacker asks for the secret key for a public key \(X\).
- sk_reveal^P(\Psi): The attacker can ask for the session key of a session \(\Psi\).
- establish_certificate(N, pk_N): The attacker can register new long-term public keys \(pk_N\) for unused identities \(N\).

The challenger maintains several variables for each party \(P\):
– **params:** Public parameters for the AKE protocol.
– **ake\_st\^{P}(\Psi):** Stores the key exchange state for the party \(P\) in the session \(\Psi\). It contains ephemeral keys that will be deleted after the completion of the key exchange.
– **result\_st\^{P}(\Psi):** Stores the resulting state for the party \(P\) for a completed session \(\Psi\). This result state contains the established session key \(k\), the identity of the peer party, which is \(\oplus\) if the peer is anonymous, otherwise the identity \(N\) of the peer. A state \(st\) that typically contains two vectors \(v_1, v_2\) that contain the ephemeral and the long-term public keys used for establishing the session key of \(\Psi\).

The concept of partner to a ephemeral public key \(X\) is introduced in the 1W-AKE security model. The attacker is a partner of a public key \(X\) if one of the following conditions hold:

– \(X\) has not been used yet.
– \(X\) is the public key that the attacker registered using the query `establish_certificate(\(N, X\))`.
– \(X\) was the response of a `send\^{P}` or `reveal\_next\^{P}` query and there is a successive query `partner\^{P}(X)`.

In order to prevent the attacker from trivially winning the game, Goldberg et al. [25] propose the freshness notion for the challenge session. A challenge session is 1W-AKE fresh if the following conditions hold:

1. Let \((k, N, st) = \text{result\_st}\^{P}(\Psi)\). For every vector \(v_1\) in \(st\) there is at least one element \(X\) in \(v_1\) such that the attacker is not a partner of \(X\).
2. If \(\text{ake\_st}\^{P}(\Psi) = (\mathcal{T}, N)\) for the challenge session \(\Psi\), the adversary did not issue `sk\_reveal\^{\Psi}(\Psi')`, for any \(\Psi'\) such that \(\text{ake\_st}\^{\Psi}(\Psi') = (\mathcal{T}, \oplus)\).

After a successful key exchange with a party \(N\), an anonymous client outputs a tuple \((k, N, v_0, v_1)\), where \(k\) is the session key. \(v_0, v_1\) is the transcript of the protocol. \(N\) outputs \((k, \oplus, v_0, v_1)\) to denote that the peer party is anonymous.

**Definition 4** (1W-AKE security). Let \(\lambda\) be a security parameter and let the number of parties \(m \geq 1\). A protocol \(\pi\) is said to be 1W-AKE secure if, for all probabilistic polynomial time (ppt) adversaries \(A\), the advantage \(Adv_{\lambda}^{\text{1w-ake}}(\pi, \lambda, m)\) that \(A\) distinguishes a session of a 1W-AKE fresh session from a randomly chosen session key is negligible in \(\lambda\), where \(Adv_{\lambda}^{\text{1w-ake}}(\pi, \lambda, m)\) is defined as:

\[
Adv_{\lambda}^{\text{1w-ake}}(\pi, \lambda, m) = |Pr(A(\text{trans}(\pi), k) = 1) - Pr(A(\text{trans}(\pi), k') = 1)|,
\]

where `trans(\pi)` is the transcript of the protocol, \(k\) is the real session key and \(k'\) is the random session key.

**Forward Secrecy.** In a key exchange protocol forward secrecy ensures that a session key derived from long-term keys remains secret even if the long-term keys are compromised in the future. A 1W-AKE secure protocol gives forward secrecy if the long-term public keys of the participating parties appear in the output vector of the protocol [25]. As in that case the adversary can be partner with a long-term public key, ensuring forward secrecy in the security game.

**Type of Adversary.** To analyze the 1W-AKE security of our protocol, we consider three types of 1W-AKE adversaries. We classify the type of adversary depending on the power of the adversary in the test session. For all other sessions the adversary can be partner to any public values, after respecting the freshness condition of the 1W-AKE security game.

**Type-I adversary.** The first type of adversary cannot be partner to any of the public values in the test session. By proving security against this kind of adversary we show that an active adversary without the knowledge of any secret values used in the test session cannot learn anything about the session key.

**Type-II adversary.** The second type of adversary can only be the partner with the ephemeral public key from a node \(N\) in the test session. By proving the security against this kind of adversary we give the security guarantee of the protocol against a man-in-the-middle adversary trying to impersonate the node \(N\) to the client.

**Type-III adversary.** The third type of adversary can only be partner with the long term public key in the test session. This gives the guarantee of forward security of the protocol; i.e., even if some information about the long-term private key is known to the adversary, the adversary cannot learn any information about the already created session key.

Due to space constraints, we shift the formal definition of 1W-anonymity to Appendix A.

4 **Our Protocol**

In this section we describe the HybridOR protocol, a hybrid lattice-based onion routing protocol. We call this protocol hybrid as here the long-term part of the key comes from a Diffie-Hellman like construction, whereas the ephemeral part of the key comes from a lattice based construction. Hence the security of the protocol essentially depends on the hard problems from both worlds, namely the hardness of the GDH problem from the world of discrete log problems and the ring-LWE problem from the world of lattice problems.
Figure 3 gives an overview of HybridOR protocol. The client generates fresh ephemeral keys $p_C \in \mathbb{R}_q$ and $g^x \in \mathbb{G}$ and sends them to the server. The server generates a fresh ephemeral key $p_N \in \mathbb{R}_q$ and computes $k_1 = p_C r_N + \epsilon' N = p_C r_N + \epsilon' N \approx ar C r_N$, a signal value $\alpha = h^R(k_1 N)$. The server sends $p_N$ and $\alpha$ to the client. The client computes $k_1 C = p_N r_C + \epsilon C'$, $k_1 = f^R(k_1 C, \alpha), k_2 = g^{\epsilon x}$ and $sk = H_1(k_1) \oplus H_2(k_2)$. From that both compute the session key $sk = H_1(k_1) \oplus H_2(k_2)$.

4.1 HybridOR Construction

The node $N$ runs the SetUp algorithm to generate the system parameters. In HybridOR the SetUp algorithm can be seen as a combination of two separate SetUp algorithms. One part generates the system parameters for the Diffie-Hellman-like key exchange (as in [3][25]) and the other part generates the parameters for the lattice based settings (as in [14]).

The SetUp algorithm generates a group $\mathbb{G}$ with large prime order $p$, where the GDH [36] problem is hard. Let $g \in \mathbb{G}$ be the generator of the group. The SetUp algorithm further generates the public parameters for the lattice based settings as described in Section 2.3. It publishes the dimension $n$, the prime modulus $q$, the description of the ring $\mathbb{R}$ and the error distribution $\chi$ in the public parameter.

The node samples $a \leftarrow \mathbb{R}$ and $s \leftarrow \mathbb{Z}_q^*$. It computes $g^s$ and publishes $(\lambda, \mathbb{R}, n, q, t, \chi, a, G, g, g^s)$ as the public parameter of the protocol, where $\lambda$ is the security parameter and $g^s$ is the long term public key of the node with $s$ as the secret key. The node also publishes $H_{st}(\cdot), H_1(\cdot), H_2(\cdot)$ and a PRF($\cdot$) in the public parameter, where $H_{st}(\cdot)$ is a collision-resistant hash function (instantiated with SHA2), $H_1(\cdot)$ is a randomness extractor (instantiated with HMAC) and $H_2(\cdot)$ is a random oracle (instantiated with SHA2). The PRF($\cdot$) is a pseudorandom function that is used to generate the key confirmation message. We instantiate the PRF($\cdot$) with AES-128. Note that according to [19], we can instantiate a randomness extractor with HMAC. However, we can also use the key derivation function HKDF [30] to instantiate $H_1$.

To initiate a new key exchange session the anonymous client $C$ calls the Init algorithm. Init randomly samples $r_C$ and $\epsilon_C$ from the error distribution $\chi$ and $x$ from $\mathbb{Z}_q^*$. It computes the ephemeral key pair as $pk_C = (p_C, g^s)$ and $sk_C = (r_C, x)$, where $p_C = ar_C + \epsilon_C \mod q$. Init sets the local session identifier as $\psi_C = H_{st}(pk_C, g^s)$, where $H_{st}$ is a collision-resistant hash function. The session information of the client is stored in the variable $st(\psi)$ as $s(t(\psi_C) = \text{(HybridOR, N, r_C, p_C, x, g^s)})$. Init generates the outgoing message $m_C = \text{(HybridOR, N, p_C, g^s)}$, and sends $(\psi_C, m_C)$ to the node $N$ over the network.

In response to the message the node runs Resp. Resp verifies whether $p_C \in \mathbb{R}_q$ and $g^s \in \mathbb{G}$. On successful verification it randomly samples $r_N$ and $\epsilon_N$ from the error distribution $\chi$ and computes $p_N = ar_N + \epsilon_N \mod q$. Resp outputs the ephemeral key pair $(p_N, r_N)$, where $p_N$ is the public part and $r_N$ remains secret to the node. Resp further samples $\epsilon' N \leftarrow \chi$ and computes $k_1 N = p_C r_N + \epsilon' N \mod q$ and $\alpha = h^R(k_1 N)$. $h^R(\cdot)$ is a randomized algorithm used to generate the signal value $\alpha$, as described in section 2.3. To ensure the correctness of the shared secret computation, $N$ sends $\alpha$ to the client [14]. The node computes the short-term shared se-

\[
\begin{align*}
\text{Client} & \quad (\text{no long-term key}) \\
\text{Node} & \quad (\text{long-term keys (s, g^s)}) \\
r_C, \epsilon_C, e_C & \leftarrow R \chi, x \leftarrow R \mathbb{G} \\
p_C = ar_C + te_C & \quad \rightarrow p_C, g^s \\
\end{align*}
\]
ing (sk_m, sk) = H_1(k_1, p_C, p_N, N, HybridOR) \oplus H_2(k_2, g^x, g^s, N, HybridOR)
\quad t_N = PRF(sk_m, N, p_N, \alpha, p_C, g^x, HybridOR, server).
Lemma 6. If $H_1$ is a randomness extractor and $H_2$ is a random oracle, no PPT Type-I adversary can distinguish between $G_0$ and $G_1$ under the GDH and the decision ring-LWE assumption.

Proof. If there exists a PPT Type-I adversary $A$ that can distinguish between $G_0$ and $G_1$, then we can construct a PPT reduction algorithm $B_{0,1}$ that can solve the GDH challenge efficiently and also can distinguish between tuples from a ring-LWE distribution and a uniform distribution.

In $G_0$, $(a, p_C)$ are samples from a ring-LWE distribution, such that $p_C = ar_C + tc_C$. In $G_1$, $(a, p_C)$ are samples from a uniform distribution over $R_q^2$, $R_q$. Under the decisional ring-LWE assumption these two distributions are indistinguishable.

To simulate the 1W-AKE challenger for $A$ the reduction algorithm $B_{0,1}$ guesses $\psi_i$ to be the test session. In the test session it takes $(G, g, g^a, g^e)$ from a GDH challenger and embeds those values in place of $(G, g^a, g^e)$. $B_{0,1}$ also takes a pair $(a_0, u_0)$ from the ring-LWE challenger and sets $a = a_0$ and $p_C = u_0$. Now if $(a_0, u_0)$ is a ring-LWE sample, then there exists an $r_C, e_C \in \chi$ such that $p_C = ar_C + te_C$ and in that case the output of $B_{0,1}$ is distributed exactly as in $G_0$. Whereas if $(a_0, u_0)$ is sample from a uniform distribution over $R_q^2$, $B_{0,1}$ simulates $G_1$ for $A$. Thus, if $A$ can distinguish $G_0$ from $G_1$, then $A \circ B_{0,1}$ can distinguish ring-LWE samples from samples from a uniform distribution over $R_q$.

It is important to note that in order to win in both cases the adversary $A$ has to query the random oracle $H_2(\cdot)$ on the same $(Z, g^x, g^e, N, \text{HybridOR})$ such that $Z = g^{xu} = g^{au}$. Whenever $A$ makes a query for some $Z \in G$ for the same $g^x$ and $g^e$, $B_{0,1}$ asks the DDH oracle whether $(g^x, g^e, Z)$ is a valid DDH tuple. If that is
the case, then $Z = g^{uv}$ and $B_{0,1}$ sends the answer to the GDH challenger. Thus if $A$ can distinguish $G_0$ from $G_1$, the $A \circ B_{0,1}$ can solve both the decision ring-LWE problem and the GDH problem. Hence,

$$\Pr(E_0) - \Pr(E_1) \leq \min(Adv^{drw}_{\mathcal{A}_o B_{1,2}}, Adv^{GDH}_{\mathcal{A}_o B_{1,2}}).$$

(2)

Lemma 7. If $H_1$ is a randomness extractor and $H_2$ is a random oracle, no PPT Type-I adversary can distinguish between $G_1$ and $G_2$ under the GDH and the decision ring-LWE assumption.

Proof. If there exists a PPT Type-I adversary $A$ that can distinguish between $G_1$ and $G_2$, then we can construct an PPT reduction algorithm $B_{1,2}$ that can solve the GDH challenge efficiently and also can distinguish between tuples from a ring-LWE distribution and a uniform distribution.

In $G_1$, $(a, p_N)$ are samples from a ring-LWE distribution, such that $p_N = a r_N + t e_N$. In $G_2$, $(a, p_N)$ are samples from a uniform distribution over $\mathbb{R}_q \times \mathbb{R}_q$. Under the decisional ring-LWE assumption these two distributions are indistinguishable. In $G_2$, $k_1$ is also distributed as a random element from $\mathbb{R}_q$. In both the cases $p_C$ is uniformly distributed over $\mathbb{R}_q$.

To simulate the 1W-AKE challenger for $A$ the reduction algorithm $B_{1,2}$ guesses $\psi_i$ to be the test session. In the test session it takes $(G, g, g^a, g^b)$ from a GDH challenger and embeds those values in place of $(G, g, g^x, g^y)$. $B_{1,2}$ also takes $\{(a_0, u_0), (a_1, u_1)\}$ from the ring-LWE challenger and sets $a = a_0$, $p_C = a_1$, $p_N = u_0$ and $k_1 = u_1$. Now if $\{(a_0, u_0), (a_1, u_1)\}$ are ring-LWE samples, then there exist $r_N, e_N, e'_N \in \chi$ such that $p_N = a r_N + t e_N$ and $k_1 = p_C r_N + t e'_N$. In that case the output of $B_{1,2}$ is distributed exactly as in $G_1$. Whereas if $\{(a_0, u_0), (a_1, u_1)\}$ are samples from uniform distribution over $R_q^2$, $B_{1,2}$ simulates $G_2$ for $A$. Thus, if $A$ can distinguish $G_1$ from $G_2$, then $A \circ B_{1,2}$ can distinguish ring-LWE samples from samples from uniform distribution over $R_q^2$.

It is important to note that in order to win in both the cases the adversary $A$ has to query the random oracle $H_2(\cdot)$ on some $(Z, g^x, g^y, N, \text{HybridOR})$ such that $Z = g^{xz} = g^{uv}$. Whenever $A$ makes an query for some $Z \in G$ for the same $g^x$ and $g^y$, $B_{1,2}$ asks the DDH oracle whether $(g^x, g^y, Z)$ is a valid DDH tuple. If that is the case, then $Z = g^{uv}$ and $B_{1,2}$ sends the answer to the GDH challenger. Thus if a PPT Type-I adversary $A$ can distinguish between $G_1$ and $G_0$, then we can construct a reduction $B_{1,2}$ which can solve both the ring-LWE and the GDH problem. As a result we can write

$$|\Pr(E_1) - \Pr(E_2)| \leq \min(Adv^{drw}_{\mathcal{A}_o B_{1,2}}, Adv^{GDH}_{\mathcal{A}_o B_{1,2}}).$$

(3)

Analysis of $G_2$. In $G_2$ the adversary has to guess a $b^*$ in the 1W-AKE game to distinguish between the real session key $sk$ and randomly chosen session key $sk'$. As $pc, p_N$ and $k_1$ are chosen uniformly at random from $\mathbb{R}_q$, and $H_1(\cdot)$ is a randomness extractor, the resulting session key $sk$ is uniformly distributed over the key space. On the other hand, $sk'$ is also chosen uniformly from the key space. As a result, the adversary has no information about $b^*$, and hence

$$\Pr(E_2) = 1/2.$$  

(4)

Conclusion. By combining equation $(1) - (4)$, we prove the result.

It is important to note that the Type-I adversary $A$ cannot be partner to any of the public values in the test session only. For all other sessions it can be a partner to almost all values after respecting the freshness criterion. So in order to simulate a 1W-AKE challenger for the adversary $A$ the reduction perfectly simulates all the other sessions. Hence the challenger can satisfy any kind of queries from $A$, and as a result perfectly simulates the game for the adversary.

5.2 Security against Type-II Adversary

Theorem 8. The protocol HybridOR is 1W-AKE secure against a PPT Type-II adversary under the GDH assumption in the random oracle model.

Proof. If there exists a PPT Type-II adversary $A$ that can break the 1W-AKE security of the protocol, then we can construct a PPT reduction algorithm $B$ against the GDH challenger. $A$ is allowed to make a polynomial number $(poly(\lambda))$ of session queries. $B$ also simulates the random oracle $H_2$. Let $P = \{P_1, \ldots, P_m\}$ be the set of parties. Let $(G, g, g^a, g^b)$ be the GDH challenge. $B$ has to compute $g^{uv}$ in order to win the GDH game.

The reduction algorithm $B$ guesses $\psi_i$ to be a test session. To simulate the $\psi_i$, $B$ runs the SetUp and generates $(\mathbb{R}, n, q, t, \chi)$. It uses the group $G$ and generator $g$ from the GDH challenger in the public parameters. $B$ samples $a \leftarrow \mathbb{R}$ sets $(a, g^u)$ as the static key pair of the server and simulates $\psi_i$ session by setting:

$$g^x = g^u,$$

$$(pc)_i = ar_C + te_C, (p_N)_i = ar_N + te_N,$$

$$(K_1)_i = (pc)_r + te'_N, (\alpha)_i = Signal((k_1)_i),$$

where, $r_C, r_N, e_C, e'_N \in \chi$.

$B$ tosses a coin and chooses $b \in F \{0,1\}$. If $b = 0$ then $B$ computes the session key by computing $H_1((k_1)_i, (pc)_i, (p_N)_i, N, \text{HybridOR}) \oplus H_2(b, g^x, g^u, N, \text{HybridOR})$, where $H_1(\cdot)$ is a randomness extractor and $B$ programs $H_2(\cdot)$ as a random oracle.
B sends the session key to A. If b = 1 then B sends a random session key to A.

A can be partner with ephemeral key \((P_N)_i\) of the server in the test session. In that case the reduction B can answer with the correct \(r_N\). With that information A can compute \(H_1((k_1)_i,(p_C)_i,(p_N)_i,N,\text{HybridOR})\). But in order to compute the correct test session key and to win the game, A has to query the random oracle \(H_2(\cdot)\) with the same input. Note that otherwise A cannot distinguish a real session key from a random one, as \(H_2(\cdot)\) is modeled as a random oracle. Whenever A makes a query \(H_3(Z,g^x,g^w,N,\text{HybridOR})\) for some \(Z \in G\), B asks the DDH oracle whether \((g^x,g^w,Z)\) is a valid DDH tuple. If that is the case, then \(Z = g^w\) and B sends the answer to the GDH challenger. Clearly the reduction \(B\) is efficient. B has to guess the test session with probability \(1/poly(\lambda)\), so if A breaks the 1W-AKE protocol with non-negligible probability then B will be able to solve the GDH problem with significant probability.

5.3 Security against Type-III Adversary

Theorem 9. The HybridOR protocol is 1W-AKE secure against a PPT Type-III adversary under the ring-LWE assumption. More precisely, for any PPT Type-I adversary \(A\),

\[
\text{Adv}_A^{1\text{w-ake}} \leq \text{Adv}_A^{\text{drLWE}} + \text{Adv}_A^{\text{drLWE}},
\]

where \(B_{0,1}\) and \(B_{1,2}\) are the reduction algorithms as described in the proof.

Proof. To prove the security against a Type-III adversary, first we define a sequence of three games \(G_0\) to \(G_2\).

\(G_0\): This is the original 1W-AKE security game, where the reduction algorithm \(B\) generates all the values honestly.

\(G_1\): This game is identical to \(G_0\), except here \(p_C\) is generated uniformly at random.

\(G_2\): This game is similar to \(G_1\), except here \(p_N\) is generated uniformly at random and also the session secret \(k_1\) is generated uniformly at random.

Lemma 10. If \(H_1\) is a randomness extractor, no PPT Type-III adversary can distinguish between \(G_0\) and \(G_1\) under the decision ring-LWE assumption.

Lemma 11. If \(H_1\) is a randomness extractor, no PPT Type-III adversary can distinguish between \(G_1\) and \(G_2\) under the decision ring-LWE assumption.

To prove Lemma 10 we consider that if there exists a PPT Type-III adversary \(A\) that can distinguish between \(G_0\) and \(G_1\), then we can construct a PPT reduction algorithm \(B_{0,1}\) that can solve the decision ring-LWE problem. Similarly for Lemma 11 if there exists a PPT Type-III adversary \(A\) that can distinguish between \(G_1\) and \(G_2\), then we can construct a PPT reduction algorithm \(B_{1,2}\) to solve the decision ring-LWE problem.

The proof of Theorem 9 is almost the same as for Theorem 5 except here the PPT reduction algorithm honestly generates \((G, g, g^x, g^z)\). So if the adversary becomes partner with the long-term public key value \(g^z\) in the test session, the reduction algorithm can satisfy the query with the correct value of \(s\).

The proof for the 1W-anonymity property (see Appendix A) of HybridOR is exactly the same as the proof of the 1W-anonymity of ntor [20]. Thus, we refer the reader for their proof and only state the result.

Lemma 12 (HybridOR is 1W-anonymous). The HybridOR protocol is 1W-anonymous in the sense of Definition 7.3

6 Performance Analysis

In this section we analyze the performance of HybridOR, and compare it with the ntor protocol employed by Tor.

6.1 Parameters

To achieve computational efficiency and to reduce the size of the public parameters, in HybridOR we use an algebraic variant of LWE called ring-LWE [33]. Similar to other ring-LWE based protocols [65949], the security and performance of HybridOR essentially depend on the three factors: \(n\), \(q\), and \(\beta\). Here, \(n\) is the degree of the irreducible polynomial \(f(x)\), \(q\) is the prime modulus and \(\beta = \sqrt{2\pi\sigma}\) for the standard deviation \(\sigma\) of the error distribution \(\chi\).

Lindner and Peikert [41] show how the parameters \((n, q, \beta)\) affect the security and performance of lattice based systems. They choose parameter set \((256, 4093, 8.35)\) for medium security level and claimed that to be comparable with 128-bit AES security. Nevertheless, several implementations of lattice-based cryptographic primitives [2245] use \(n = 512\) to achieve high security. To be on the safer side, we also choose a high security level, and use parameter set \((512, 1051649, 8.00)\) (as used in [45]) in our implementation for \(\mathbb{R}_q\).

For the DLog group \(G\), we use the elliptic curve cryptographic (ECC) setting with points (compressed form) of size \(p = 256\) bits, such as provided by Bernstein’s Curve25519 [5].

6.2 Computation Cost

We assume that the elements \(r_C, e_C, p_C\) and \(g^z\) are pre-computed on the client side, and the elements \(r_N, e_N, e_N'\),
and $p_N$ are precomputed on the node side, e.g. in idle cycles. In our analysis, they are received by the code as an input. In that case, to compute the session secret $\{k_1, k_2\}$, the client and the node each have to perform 1 multiplication and 1 addition in $\mathbb{R}_q$ and 1 exponentiation in $\mathbb{G}$.

Multiplications over $\mathbb{R}_q$ can be performed efficiently using an FFT-based algorithm [33], which takes $O(n \log n)$ for a serial implementation and $O(\log n)$ time for a parallel implementation [28]. It is important to observe that these multiplications are more efficient than exponentiation in $\mathbb{G}$ (even in ECC settings). As a result the total computation cost of the node (with precomputation) is mainly dominated by exponentiation in $\mathbb{G}$.

As a proof of concept, we implement our protocol in a machine with a 6-core Intel Xeon (W3690) processor, each core running at 3.47 GHz. We use the GMP [48] library and the Tor library to implement the protocol. The code is compiled with -O3 optimizations using gcc 4.6.3.

For our choice of parameter set $(512, 1051649, 8.00)$ and ECC Curve25519, both the client and the node require $\approx 150 \mu s$ to compute the shared secret. The multiplication along with one addition in $\mathbb{R}_q$ only requires $\approx 50 \mu s$, and the exponentiation in $\mathbb{G}$ requires $\approx 100 \mu s$.

The Tor protocol in Tor requires two exponentiations in $\mathbb{G}$ on both sides, and correspondingly requires $\approx 200 \mu s$ to compute the shared secret. As a result, our proof-of-concept HybridOR implementation is nearly 1.5 times faster than the Tor protocol used in Tor.

7 Conclusion

Lattice-based cryptographic protocols are supposed to offer resilience against attacks by quantum computers, and the recent efficient ring-based constructions also put them in the realm of the practical use. In this paper, we demonstrated their utility to onion routing. In particular, we have presented a novel lattice-based 1W-AKE protocol HybridOR, which extracts its security from both the classically secure GDH assumption and the quantum-secure ring-LWE assumption. On one hand, we base its security against man-in-the-middle impersonation attacks only on the GDH assumption as we do not expect an adversary to have quantum capabilities today, and it allows us to leverage the current Tor PKI in its current form. On the other hand, we base its forward secrecy on the the quantum-secure ring-LWE assumption, which allows us to make HybridOR more efficient compared to the currently employed ntor protocol.

We also analyzed performance of our protocol in terms of its computation and communication cost for the 128-bit security setting. Our performance analysis demonstrates that post-quantum 1W-AKE can already be considered practical for use today.

Finally, we view our efficient HybridOR construction to be of independent interest to other authenticated key exchange protocols as well as anonymous communication scenarios over the Internet, and we plan to explore some those scenarios in the future.

6.3 Communication Cost

In the HybridOR protocol the client has to send an element $g^x \in \mathbb{G}$ and an element $p_C \in \mathbb{R}_q$ to the node. We require 32 bytes to represent an element on Curve25519. The multiplication along with one addition in $\mathbb{R}_q$, and the exponentiation in $\mathbb{G}$ will require $\approx 200 \mu s$ (even in ECC settings). As a result, our proof-of-concept HybridOR implementation is nearly 1.5 times faster than the Tor protocol used in Tor.

References


Note that, for ntor, using some parallelization technique both the node and the client can reduce the computation cost to 1.33 exponentiations (for $\lambda = 128$) [3], however, the current Tor implementation does not employ these.


A One-way Anonymity

The purpose of one-way anonymity is that an adversary (even a server) cannot guess which client is participating in the key exchange. The client always knows that it is participating in a key exchange protocol with the server, but from the server’s point of view (or from the point of view of any other user), the participating client must be anonymous.

In Figure 5 the 1W-Anonymity game is explained in detail. In the security experiment the adversary can communicate with all the parties directly through a \( \text{Ch}^{\text{KE}} \) challenger. The adversary chooses two distinct party indices \( i \) and \( j \) for the key exchange challenge session \( \Psi^* \) and gives that to \( \text{Ch}^{\text{AN}} \). \( \text{Ch}^{\text{AN}} \) chooses a candidate party by picking one index \( b^* \leftarrow \{ i, j \} \) randomly, and starts a key-exchange session \( \Psi^* \) with \( P_{i^*} \). Finally the adversary has to guess \( b^* \). To prevent trivial winning the adversary can not execute certain queries \([3][20]\) that leak the state of the candidate parties. Formally, to satisfy this condition we require that \( \text{Ch}^{\text{AN}} \) internally runs a copy of 1W-AKE challenger \( \text{Ch}^{\text{KE}} \). We denote the internal copy as \( \text{ICl}_{\text{KE}} \).

**Definition 13 (1W-anonymity).** Let \( \lambda \) be the security parameter. Let \( P, N \) be ppt interactive turing machines, and \( v \) denote the view of the adversary while interacting with \( P \) and \( N \), i.e. \( v = \langle A(1^\lambda), P(1^\lambda), N(1^\lambda) \rangle \). Let \( b \) be the output of the adversary after a 1W-anonymity game. A protocol \( \pi \) is said to be 1W-anonymous if, for all ppt adversaries \( A \), the advantage \( \text{Adv}^{1\text{w-anon}}_{\pi}(\pi, \lambda, m) \) is negligible in \( \lambda \), where:

\[
\text{Adv}^{1\text{w-anon}}_{\pi}(\pi, \lambda, m) = \\
\frac{| Pr[b \leftarrow \langle A(1^\lambda), \text{Ch}^{\text{AN}}_A(1^\lambda), \text{Ch}^{\text{KE}}(1^\lambda) \rangle | b = b^*| - Pr[b \leftarrow \langle A(1^\lambda), \text{Ch}^{\text{AN}}_A(1^\lambda), \text{Ch}^{\text{KE}}(1^\lambda) \rangle | b \neq b^*]|}{Pr[b \leftarrow \langle A(1^\lambda), \text{Ch}^{\text{AN}}_A(1^\lambda), \text{Ch}^{\text{KE}}(1^\lambda) \rangle | b \neq b^*]|}.
\]

![Figure 5: The anonymizing machine \( \text{Ch}^{\text{AN}}_A(1^\lambda); \text{ICl}_{\text{KE}}(1^\lambda) \) is an internally emulated copy of \( \text{Ch}^{\text{KE}}(1^\lambda) \)[3]](http://eprint.iacr.org/)

---


