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

Satrajit Ghosh[†] Aniket Kate[‡]

[†]Indian Statistical Institute (ISI), Kolkata, India
[‡]CISPA, Saarland University, Germany
satrajitgh@gmail.com
aniket@mmci.uni-saarland.de

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 user 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 privacy of today's anonymous communication cannot be ensured once quantum computers arrive.

In this paper, we demonstrate utility of 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 or 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 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 manageable communication overhead to the Tor protocol.

Keywords: Tor, Onion routing, Forward anonymity, Learning with errors, Compatibility

1 Introduction

Lattice-based cryptographic constructions have drawn an overwhelming amount of research attention in the last decade [7, 34, 36, 39, 43]. Their strong provable worst case security guarantee, apparent resistance to quantum attacks, high asymptotic efficiency and flexibility towards realizing powerful primitives (e.g., fully homomorphic encryption [21]) have been the vital reasons behind their 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 [25,32,34]. As a result, some work has started to appear towards developing lattice-based version of real-world cryptographic protocols [6, 40, 48]. In this work, we explore the utility of plausibly 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 [11, 12, 16, 23, 41, 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 [16, 47] has turned out to be a huge success. Today, along with anonymous web browsing and service hosting, Tor is also extensively used for censorship-resistant communication [14].

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 anonymous 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 that 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 protocol called the Tor authentication protocol (TAP) to agree upon those session keys in a *telescoping (or multi-pass)* construction [16]. Due to its atypical use of CPA-secure RSA encryption, TAP was considered weaker in terms of performance as well as security [22]. Recently, Goldberg, Stebila and Ustaoglu [24] 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 [15].

Security of ntor and other 1W-AKE protocols [3, 10, 27–29] 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 confidentially and anonymity of the OR communications will be broken in the post-quantum world. Importantly, the current 1W-AKE protocols are also *not* forward-secure against the quantum attacks; the confidentially 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 forward security in the post-quantum world without significantly affecting the current cryptographic infrastructure and performance.

Our 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 or 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. Moreover, while achieving this enhanced security properties, HybridOR does not require any modifications to the current Tor public keys or 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 OR protocol [2].

2 Background

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

2.1 Onion Routing

In the original OR protocol [41] 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 [16] (i.e., Tor), circuits are constructed incrementally and interactively, where symmetric session keys are established using a forward-secure authenticated 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 provably secure (against the GDH assumption [37]) and significantly more efficient ntor protocol [24].

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 authenticated key exchange [24].

One-way Authenticated Key Exchange—1W-AKE. Goldberg et al. introduce a security definition of (one-way anonymous) one-way authenticated key exchanges (1W-AKE) to facilitate design of provably secure session key agreement protocols for onion routing [24]. (See Section 3 for a complete 1W-AKE definition.)

They also fixed a key agreement protocol proposed in [38] to obtain a provably secure construction called the ntor protocol, which has replaced the TAP protocol in the current Tor network. In ntor, 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 $\mathsf{H}((g^x)^y, (g^x)^b)$, 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 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^y 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 a multi-exponentiation trick. (See Figure 4 in Appendix A for a pictorial illustration of ntor and Ace.)

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 without regularly rotating the PKI keys for all Tor nodes is not possible [29], and periodic public key rotation should be avoided for scalability reasons. There have been attempts to solve this problem in the identity-based cryptography setting [29] or the certificate-less cryptography setting [10]. Nevertheless, as discussed in [2], key authorities required in these constructions can be difficult to implement in practice.

2.2 Gap Diffie-Hellman—GDH

Let $\mathbb{G}$ be a multiplicative group with large prime order p and $g \in \mathbb{G}$ be the generator of the group. Given a triple (g, g^a, g^b) for $a, b \in_r \mathbb{Z}_p^*$, 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 [37]. The DDH oracle $\mathcal{O}^{ddh}$ takes input as (G, g, g^a, g^b, z) for some $z \in \mathbb{G}$ and tells whether $z = g^{ab}$ or not, that is whether the tuple is a DH tuple

or not. For the security parameter λ, solving GDH problem in $\mathbb{G}$ is assumed to be a hard problem. More formally,

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

$$Adv_A^{gdh} = \Pr[A^{\mathcal{O}^{ddh}}(p, g, g^a, g^b) = g^{ab}, (a, b) \in_r \mathbb{Z}_p^{*2}].$$

The GDH assumption states that Adv_A^{gdh} is a negligible function of the security parameter λ for all PPT algorithms A.

2.3 Learning With Errors—LWE

Learning with errors (LWE) is a problem of distinguishing noisy random linear equations from truly random ones, for a small amount of noise. It has been shown to be as hard as some worst case lattice problems [43], and its different variants have been employed in designing lattice-based cryptosystems [33, 35, 42, 43]. The main drawback of schemes based on LWE [43] is that they are based on matrix operations, which are quite inefficient and result in large key sizes. To overcome these problems, in last few years, special lattices with additional algebraic structures are used to construct cryptographic protocols.

LWE for Polynomial Ring. To reduce computation, communication and storage complexity, Lyubashevsky [34] propose an algebraic variant of LWE, *ring-LWE*, the problem is defined over a polynomial ring.

Let $\mathbb{Z}_q$ be the set of integers from $\lfloor -q/2 \rfloor$ to $\lfloor q/2 \rfloor$, and let $\mathbb{Z}[x]$ be the set of polynomials with 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, which makes $f(x)$ irreducible over the $\mathbb{Z}$. Let $R = \mathbb{Z}[x]/\langle f(x) \rangle$ be the ring of integer polynomials modulo $f(x)$ such that elements of 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 $R_q = \mathbb{Z}_q[x]/\langle f(x) \rangle$ be the ring of integer polynomials modulo both $f(x)$ and q. The ring 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 $f(x)$.

Let χ be the error distribution over R, which is concentrated on *small* elements of R. See [34] for details of the error distribution for the security and the correctness of the system. We denote $\mathbb{D}_{s,\chi}$ as the ring-LWE distribution over R_q^2, obtained by choosing uniformly random $a \leftarrow R_q$ and $e \leftarrow \chi$, and outputs $(a, a \cdot r + e)$ for some $r \leftarrow R_q$.

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

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

$$Adv_A^{drlwe} = |\Pr[A^{\mathcal{O}^{\mathbb{D}_{s,\chi}}}(\cdot)] - \Pr[A^{\mathcal{O}^{U_{R_q \times R_q}}}(\cdot)]|.$$

The decision ring-LWE assumption states that for given values of n, q, and χ, for every PPT adversary A, Adv_A^{drlwe} is negligible in the security parameter λ.

The hardness results for the LWE problem are described in [34, 39, 43]. Brakerski et al. [7] show the classical hardness of the LWE problem. Ding et al. [13] mention that for any $t \in \mathbb{Z}^+$, such that $gcd(t, q) = 1$, the LWE assumption still holds if we choose $b = \langle \mathbf{a}, \mathbf{r} \rangle + te$. We use $t = 2$ for our construction.

It is important to note that ring-LWE samples are pseudorandom even when the secret r is chosen from the error distribution [1, 36]. Ducas et al. [17] show that the ring-LWE problem is hard in any ring $\mathbb{Z}[x]/\langle \Phi_m \rangle$, 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 (or reconciliation) of the shared secret. In literature, there are different methods [13, 20] to agree on a shared secret from noisy shared secret values. For our construction we adopt the method due to Ding et al. [13] and recall the corresponding concept of robust extractors and the signal functions below.
Intuition. Let Alice sends $p_A = as_A + 2e_A$ to Bob, and Bob sends $p_B = as_B + 2e_B$ to Alice, where $e_A, e_B \in \chi$ and $a, s_A, s_B \in R_q$. Here, s_A and s_B are the secret keys of Alice and Bob respectively. To compute the shared secret Alice computes $K_A = p_B s_A \mod q = as_A s_B + 2e_B s_A \mod q$ and Bob computes $K_B = p_A s_B \mod q = as_A s_B + 2e_A s_B \mod q$. Clearly $K_A - K_B$ is even and small. Using the robust extractor as explained in [13], 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 $[-q/4, q/4] \bigcap \mathbb{Z}$ or not. If K_B lies in $[-q/4, q/4] \bigcap \mathbb{Z}$ then $K_B = K_A + 2(e_A s_B - e_B s_A) \mod q$. Now $2(e_A s_B - e_B s_A) \leq q/4$, implies:

$$\begin{aligned} K_B &= K_A + 2(e_A s_B - e_B s_A) \mod q, \\ K_B &= K_A \mod q + 2(e_A s_B - e_B s_A), \\ (K_B \mod q) &\mod 2 = (K_B \mod q) \mod 2. \end{aligned}$$

This is also true when K_B lies outside the interval $[-q/4, q/4] \bigcap \mathbb{Z}$. 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. [13] 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 δ 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 $\alpha = h(y) \in \{0, 1\}$.*
- *$f(x, \alpha) = f(y, \alpha)$, 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 [13]. For $q > 2$ define $\alpha_0 : \mathbb{Z}_q \to \{0, 1\}$ and $\alpha_1 : \mathbb{Z}_q \to \{0, 1\}$ as follows:

$$\alpha_0(x) = \begin{cases} 0, & \text{if } x \in [-\lfloor \frac{q}{4} \rfloor, \lfloor \frac{q}{4} \rfloor]; \\ 1, & \text{otherwise.} \end{cases} \quad \alpha_1(x) = \begin{cases} 0, & \text{if } x \in [-\lfloor \frac{q}{4} \rfloor + 1, \lfloor \frac{q}{4} \rfloor + 1]; \\ 1, & \text{otherwise.} \end{cases}$$

The hint algorithm $h(\cdot)$ generates the signal α 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 [13], 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 R_q can be represented by a degree $n-1$ polynomial. For example any $a \in R_q$ can be written in the form $a_0 + a_1 x + \cdots + a_{n-1}x^{n-1}$. In that case the extractor can extract n bits from an element of R_q. We extend $\alpha_0^R(a), \alpha_1^R(a) : R_q \to R_2$ as follows:

$$\alpha_0^R(a) = \sum_{i=0}^{n-1} \alpha_0(a_i)x^i;\ \alpha_1^R(a) = \sum_{i=0}^{n-1} \alpha_1(a_i)x^i.$$

The algorithm $h^R(\cdot)$ can be defined in the same manner as $h^R(a) = \alpha_b^R(a)$, for $b \leftarrow \{0,1\}$. Similarly define the extractor in the ring settings $f^R(a,\alpha) : R_q \to R_2$ 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. [19] provide the first CK^+ secure [9, 19, 30] authenticated key exchange (AKE) protocol from a key-encapsulation mechanism (KEM) based on ring-LWE problem in the standard model. However due to the huge communication cost (≈ 139625 bytes) their lattice-based AKE is not suitable for real-world applications. In [20], 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 [20] reaches about 10075 bytes. Peikert [40] 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. [13] 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. [48] 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 et al. [24] 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.

A 1W-AKE protocol is a tuple of ppt algorithms $(\mathsf{SetUp}, \mathsf{Init}, \mathsf{Resp}, \mathsf{CompKey})$, where SetUp generates the system parameters and the static long-term keys for the node. The client calls Init to initiate the 1W-AKE protocol and the node uses Resp to respond to an Init. Finally, the client uses $\mathsf{CompKey}$ to verify the key-confirmation message and compute the key. We assume that a PKI is given, that means for a node N all parties $\{P_1, \cdots, P_m\}$ can obtain a certified public key pk_N.

Along with protocol correctness, a secure 1W-AKE protocol should respect the following properties:

1W-AKE security. An attacker should not learn anything about the session key of an uncompromised session, even if it completely compromises several other sessions, introduces fake identities or even learn some uncompromised session secret.

1W-anonymity. A node should not be distinguish between communicating with two different clients.

upon $\mathsf{send}^P(params, N)$**:**
$(m, st, \Psi) \leftarrow \mathsf{Init}(N, params, cs)$
$ake\_st^P(\Psi) \leftarrow (N, st)$; send (m, Ψ)

upon $\mathsf{send}^P(\Psi, m)$ **and** $ake\_st^P(\Psi) = \perp$**:**
$(m', (k, \star, st), \Psi) \leftarrow \mathsf{Resp}(sk_P, P, m, cs)$;
$result\_st^P(\Psi) \leftarrow (k, st, \star)$; send m'

upon $\mathsf{send}^P(\Psi, m)$ **and** $ake\_st^P(\Psi) \neq \perp$**:**
$(N, st) \leftarrow ake\_st^P(\Psi)$; check for a valid pk_N
$(k, N, st) \leftarrow \mathsf{CompKey}(pk_N, m, \Psi, (N, st))$
erase $ake\_st^P(\Psi)$; $result\_st^P(\Psi) \leftarrow (k, N, st)$

upon $\mathsf{reveal\_next}^P$**:**
$(x, X) \leftarrow \mathsf{Gen}(1^\lambda)$; append (x, X) to cs; send X

upon $\mathsf{partner}^P(X)$**:**
if a key pair (x, X) is in the memory **then** send x

upon $\mathsf{sk\_reveal}^P(\Psi)$**:**
if $result\_st^P(\Psi) = (k, N, st)$ **then** send k

upon $\mathsf{establish\_certificate}(N, pk_N)$**:**
register the public key pk_N for the party N

upon $\mathsf{test}^P(\Psi)$**:** *(one time query)*
$(k, N, st) \leftarrow result\_st^P(\Psi)$
if $k \neq \perp$ and $N \neq \star$ and Ψ is 1W-AKE fresh **then**
if $b = 1$ **then** send k **else** send $k' \leftarrow_R \{0,1\}^{|k|}$

Figure 1: 1W-AKE Security Challenger: $\mathsf{Ch}_b^{\mathrm{KE}}(1^\lambda)$, where λ is the security parameter. If any invocation outputs $\perp$, the challenger erases all session-specific information for that session and aborts that session. [3]

3.1 Correctness

In a 1W-AKE protocol an anonymous client (denoted by $\circledast$) tries to establish a shared secret key with a node N. The client calls $\mathsf{Init}(N, pk_N, cs)$, which returns an output message m, session id Ψ 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 $\mathsf{Resp}(sk_N, N, m, cs)$ and outputs $(m', (k, \circledast, \overrightarrow{v}), \Psi_N)$, where m' is the response message to the client, k is the session key computed by N, and $\overrightarrow{v}$ contains ephemeral public keys for the session Ψ_N. On receiving m', the client computes $(k', N, \overrightarrow{v}')$ by calling $\mathsf{CompKey}(pk_N, m', \Psi, st)$, where k' is the session key computed by the client and $\overrightarrow{v}'$ is the list of ephemeral public keys. An 1W-AKE protocol is correct if for every party N:

$$\Pr[(m, st, \Psi) \leftarrow \mathsf{Init}(N, pk_N, cs), (m', (k, \circledast, \overrightarrow{v}), \Psi_N) \leftarrow \mathsf{Resp}(sk_N, N, m, cs), \\ (k', N, \overrightarrow{v}') \leftarrow \mathsf{CompKey}(pk_N, m', \Psi, st) : k = k' \wedge \overrightarrow{v} = \overrightarrow{v}')] = 1.$$

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. It requires an active attacker to not learn anything about the key or be able to impersonate an honest node.

In the security game, a challenger $\mathsf{Ch}^{\mathrm{KE}}$ represents honest parties $(P_1, \cdots, P_m)$ and allows the attacker a fixed set of queries described in Figure 1. The challenger internally runs the 1W-AKE algorithm, and simulates each party. For the challenge, the adversary asks $\mathsf{Ch}^{\mathrm{KE}}$ for the session key of an uncompromised session Ψ for a party P by querying $test^P(\Psi)$ (one time query). $\mathsf{Ch}^{\mathrm{KE}}$ sends the correct session key or a randomly chosen session key to the attacker with equal probability. The attacker's task is to determine whether the given key corresponds to the real session Ψ or is random.

For triggering the initiation session, triggering the response to a key exchange, and for completing a key exchange, the challenger allows the adversary to query $\mathsf{send}^P(\cdot, m)$. For the compromising parties, the attacker can ask the following queries:

- $\mathsf{reveal\_next}^P$: ask the party P to reveal the next public key that will be chosen.
- $\mathsf{partner}^P(X)$: ask for the secret key for a public key X.

– $\mathsf{sk\_reveal}^P(\Psi)$: ask for the session key of a session Ψ.
– $\mathsf{establish\_certificate}(N, pk_N)$: register new long-term public keys pk_N for an unused identity N.

The challenger maintains several variables for each party P:

– $params$ stores public parameters for the AKE protocol.
– $ake\_st^P(\Psi)$ stores the key exchange state for the party P in the session Ψ. 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 Ψ. This result state contains the established session key k, the identity of the peer party, which is $\circledast$ if the peer is anonymous, otherwise the identity of the peer. A state st that typically contains two vectors $\overrightarrow{v_0}, \overrightarrow{v_1}$ that contain the ephemeral and the long-term public keys used for establishing the session key of Ψ.

The attacker is a $\mathsf{partner}$ 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 a $\mathsf{establish\_certificate}(N, X)$ query.
– X was the response of a send^P or $\mathsf{reveal\_next}^P$ query and there is a successive query $\mathsf{partner}^P(X)$.

In order to prevent the attacker from trivially winning the game, Goldberg et al. [24] propose the *freshness* notion for the challenge session. A challenge session is *1W-AKE fresh* if the following conditions hold:

1. Let $(k, N, st) = result\_st^P(\Psi)$. For every vector $\overrightarrow{v_i}$ in st there is at least one element X in $\overrightarrow{v_i}$ such that the attacker is not a $\mathsf{partner}$ of X.
2. If $ake\_st^P(\Psi) = (\overrightarrow{v}, N)$ for the challenge session Ψ, the adversary did not issue $\mathsf{sk\_reveal}^N(\Psi')$, for any Ψ' such that $ake\_st^N(\Psi') = (\overrightarrow{v}, \circledast)$.

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

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

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

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 key exchange 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 provides forward secrecy if the long-term public keys of the participating parties appear in the output vector of the protocol [24]. In that case the adversary can be $\mathsf{partner}$ with a long-term public key, ensuring forward secrecy in the security game.

3.3 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 view of any other user), the participating client must be anonymous.

In Figure 2 the 1W-Anonymity game is explained in detail. In the security experiment the adversary can communicate with all the parties directly through a Ch^{KE} challenger. The adversary chooses two distinct party indices i and j for the key exchange challenge session Ψ^* and gives that to Ch^{AN}. Ch^{AN} chooses a candidate party by picking one index $b^* \leftarrow \{i, j\}$ randomly, and starts a key-exchange session Ψ^* with P_{b^*}. Finally the adversary has to guess b^*. To prevent trivial winning the adversary can not execute certain queries [3, 24] that leak the state of the candidate parties. Formally, to satisfy this condition we require that Ch^{AN} internally runs a copy of 1W-AKE challenger Ch^{KE}. We denote the internal copy as ICh^{KE}.

Definition 5 (1W-anonymity)**.** *Let λ 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 π is said to be 1W-anonymous if, for all ppt adversaries A, the advantage $Adv_A^{\textit{1w-anon}}(\pi, \lambda, m)$ is negligible in λ, where:*

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

upon $\mathsf{start}(i, j, params, N)$**:** *(one time query)*
  if $i \neq j$ **then**
    if $b = 1$ **then** $i^* \leftarrow i$ **else** $i^* \leftarrow j$
    send $\mathsf{send}^{P_{i^*}}(params, N)$ to $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$
    wait for the response (Ψ^*, m'); send m' to $\mathcal{M}$

upon $\mathsf{send}(m)$**:**
  forward $\mathsf{send}^{P_{i^*}}$ to $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$

upon reveal_next**:**
  forward $\mathsf{reveal\_next}^{P_{i^*}}$ to $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$

upon sk_reveal**:**
  forward $\mathsf{sk\_reveal}^{P_{i^*}}(\Psi^*)$ to $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$

upon $\mathsf{partner}(X)$**:**
  forward $\mathsf{partner}^{P_{i^*}}(X)$ to $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$

Figure 2: The anonymizing machine $\mathsf{Ch}_b^{\text{AN}}(1^\lambda)$: $\mathsf{ICh}_1^{\text{KE}}(1^\lambda)$ is an internally emulated copy of $\mathsf{Ch}_1^{\text{KE}}(1^\lambda)$ [3]

4 Our Protocol

In this section we describe the HybridOR protocol, a hybrid lattice-based onion routing protocol. We call this protocol hybrid as the long-term part of the key comes from a DH key exchange, whereas the ephemeral part of the key comes from a lattice based key exchange. Hence the security of the protocol essentially depends on the hard problems in either setting, namely the hardness of the GDH problem from the DLog setting or the hardness of the ring-LWE problem from the lattice-based setting.

In our HybridOR protocol, The client generates fresh ephemeral keys $p_C \in R_q$ and $g^x \in \mathbb{G}$ and sends them to the node. The node generates a fresh ephemeral key $p_N \in R_q$ and computes $k_{1N} = p_C r_N + te'_N \approx ar_C r_N$, a signal value $\alpha = h^R(k_{1N})$. The node sends p_N and α to the client. The client computes $k_{1C} = p_N r_C + tr'_C \approx ar_C r_N$. Recall that $t = 2$. The client and node *approximately agree* on the shared secret value k_{1C} and k_{1N}. To achieve *exact agreement* on the shared secret from the approximate shared secret, the robust extractor $f^R(\cdot)$ is used. The client and node compute the shared secret k_1, k_2, and sk as follows:

$$k_1 = f^R(k_{1N}, \alpha),\ k_2 = (g^x)^s,\ sk = H_1(k_1) \oplus H_2(k_2) \text{ (node-side)}$$
$$k_1 = f^R(k_{1C}, \alpha),\ k_2 = (g^s)^x,\ sk = H_1(k_1) \oplus H_2(k_2) \text{ (client-side)}$$

$\mathsf{SetUp}(N, \lambda)$:

1. For the security parameter λ, generate system parameters (R, n, q, t, χ) and $(\mathbb{G}, g, p)$.
2. Sample $a \leftarrow R_q$.
3. Sample $s \leftarrow \mathbb{Z}_p^*$ and compute g^s.
4. Output (a, g^s) as public key, and s as secret key.

$\mathsf{Init}((a, g^s), N)$:

1. Sample $(r_C, e_C) \in \chi$ and $x \leftarrow \mathbb{Z}_p^*$.
2. Generate ephemeral key pairs $(r_C, p_C = ar_C + te_C)$ and (x, g^x).
3. Set session id $\Psi_C \leftarrow H_{st}(p_C, g^x)$.
4. Update $st(\Psi_C) \leftarrow (\mathsf{HybOR}, N, r_C, p_C, x, g^x)$.
5. Set $m_C \leftarrow (\mathsf{HybOR}, N, p_C, g^x)$.
6. Output m_C , Ψ_C.

$\mathsf{Resp}((a, g^s), s, p_C, g^x)$:

1. Sample $(r_N, e_N, e'_N) \in \chi$.
2. Generate an ephemeral key pair $(r_N, p_N = ar_N + te_N)$.
3. Compute $k_{1N} = p_C r_N + te'_N$ and $\alpha = h^R(k_{1N})$.
4. Set session id $\Psi_N \leftarrow H_{st}(p_N, \alpha)$.
5. Compute $k_1 = f^R(k_{1N}, \alpha)$ and $k_2 = (g^x)^s$.
6. Compute $(sk_m, sk) \leftarrow H_1(k_1, p_C, p_N, N, \mathsf{HybOR}) \oplus H_2(k_2, g^x, g^s, N, \mathsf{HybOR})$.
7. Compute $t_N = PRF(sk_m, N, p_N, \alpha, p_C, g^x, \mathsf{HybOR}, \mathsf{node})$.
8. Set $m_N \leftarrow (\mathsf{HybOR}, p_N, \alpha, t_N)$.
9. Erase r_N and output m_N.

$\mathsf{CompKey}((a, g^s), \Psi_C, t_N, p_N, \alpha)$:

1. Retrieve N, r_C, p_C, x, g^x from $st(\Psi_C)$ if it exists.
2. Compute $k_{1C} = p_N r_C + te_C$.
3. Compute $k_1 = f^R(k_{1C}, \alpha)$ and $k_2 = (g^s)^x$.
4. Compute $(sk_m, sk) \leftarrow H_1(k_1, p_C, p_N, N, \mathsf{HybOR}) \oplus H_2(k_2, g^x, g^s, N, \mathsf{HybOR})$.
5. Verify $t_N = PRF(sk_m, N, p_N, \alpha, p_C, g^x, \mathsf{HybOR}, \mathsf{node})$.
6. Erase $st(\Psi_C)$ and output sk.

If any verification fails, the party erases all session-specific information and aborts.

Figure 3: A detailed description of the HybridOR protocol

4.1 Construction

Figure 3 provides a detailed description of the HybridOR protocol. 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 DH-like key exchange (as in [3, 24]) and the other part generates the parameters for the lattice based settings (as in [13]).

The SetUp algorithm generates a group $\mathbb{G}$ with large prime order p, where the GDH [37] 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 R and the error distribution χ in the public parameter.

The node samples $a \leftarrow R_q$ and $s \leftarrow \mathbb{Z}_p^*$, computes g^s, and publishes $(R, n, q, t, \chi, a, G, g, g^s)$ as the public parameter of the protocol. where 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, $H_1(\cdot)$ is a randomness extractor and $H_2(\cdot)$ is a random oracle. The $PRF(\cdot)$ is a pseudorandom function that is used to generate the key confirmation message. Note that according to [18], we instantiate a randomness extractor with HMAC. However, we can also use the key derivation function HKDF [31] 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 e_C from the error distribution χ and x from $\mathbb{Z}_p^*$. It computes the ephemeral key pair as $pk_C = (p_C, g^x)$ and $sk_C = (r_C, x)$, where $p_C = ar_C + te_C \mod q$. Init sets the local session identifier as $\psi_C = H_{st}(p_C, g^x)$, where H_{st} is a collision-resistant hash function. The session information of the client is stored in the variable $st(\psi)$ as $st(\psi_C) = (\mathsf{HybOR}, N, r_C, p_C, x, g^x)$. Init generates the outgoing message $m_C = (\mathsf{HybOR}, N, p_C, g^x)$, and sends (ψ_C, m_C) to the node N over the network.

In response to the message the node runs Resp, which verifies whether $p_C \in R_q$ and $g^x \in \mathbb{G}$. On successful verification it randomly samples r_N and e_N from the error distribution χ and computes $p_N = ar_N + te_N \mod q$. 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 $e'_N \leftarrow \chi$ and computes $k_{1N} = p_C r_N + te'_N \mod q$ and $\alpha = h^R(k_{1N})$. $h^R(\cdot)$ is a randomized algorithm used to generate the signal value α, as described in section 2.3. To ensure the correctness of the shared secret computation, N sends α to the client [13]. The node computes the short-term shared secret (k_1) and the long-term shared secret (k_2) as:

$$k_1 = f^R(k_{1N}, \alpha),\ k_2 = (g^x)^s = g^{xs},$$

where $f^R(\cdot)$ is the robust extractor as defined in Section 2.3. By short-term shared secret we mean the shared secret computed using the client's ephemeral key and node's ephemeral key. By long-term shared secret we mean the shared secret computed by using the client's ephemeral key and node's long-term or static key.

The node computes the session key sk, the PRF key sk_m and the key confirmation message t_N as:

$$\begin{aligned}(sk_m, sk) =& H_1(k_1, p_C, p_N, N, \mathsf{HybOR}) \oplus H_2(k_2, g^x, g^s, N, \mathsf{HybOR})\\ t_N =& PRF(sk_m, N, p_N, \alpha, p_C, g^x, \mathsf{HybOR}, \mathsf{node}).\end{aligned}$$

The tag t_N provides only a means for the key confirmation. Resp returns the session identifier $\psi_N = H_{st}(p_N, \alpha)$ and a message $m_N = (\mathsf{HybOR}, p_N, \alpha, t_N)$. The node sends (ψ_N, m_N) to the client. The node completes the session by deleting (r_N, e_N, e'_N) and outputting $(sk, \circledast, (\overrightarrow{v_0}, \overrightarrow{v_1}))$, where $\overrightarrow{v_0} = \{p_C, g^x\}$ and $\overrightarrow{v_1} = \{p_N, g^s\}$. $\circledast$ denotes that the identity of the client is not known to the node.

On receiving the message (ψ_N, m_N) for the session ψ_C, the client C calls the algorithm CompKey to compute the session key. CompKey first checks whether the session ψ_C is active; if so, it retrieves the required session information, namely r_C, p_C, x, g^x from $st(\psi_C)$. Then it checks whether $p_N \in R_q$. After successful verification CompKey computes the shared secrets k_1, k_2 as follows:

$$\begin{aligned}k_{1C} &= p_N r_C + te_C \mod q,\\ k_1 &= f^R(k_{1C}, \alpha),\ k_2 = (g^s)^x = g^{xs}.\end{aligned}$$

The client computes $(sk_m, sk) = H_1(k_1, p_C, p_N, N, \mathsf{HybOR}) \oplus H_2(k_2, g^x, g^s, N, \mathsf{HybOR})$, where sk is the session key and sk_m is the PRF key. It verifies the key-confirmation message t_N using the key sk_m. After that the client completes the session ψ_C by deleting $st(\psi_C)$ and outputting $(sk, N, (\overrightarrow{v_0}, \overrightarrow{v_1}))$, where $\overrightarrow{v_0} = \{p_C, g^x\}$ and $\overrightarrow{v_1} = \{p_N, g^s\}$. If any verification fails during the session execution, the party erases all session-specific information and aborts the session.

In Figure 4 in Appendix A, we compare HybridOR with the ntor and Ace protocols in the literature.

Correctness. To analyze the correctness of HybridOR, we can see HybridOR as a combination of two key exchange protocols, namely the Diffie-Hellman key exchange protocol and the lattice-based protocol by

by Ding et. al [13]. Hence the correctness of HybridOR directly follows from the correctness of DH key exchange and the correctness of the lattice-based protocol [13].

For the DH part, the node computes $(g^x)^s = g^{xs}$ and the client computes $(g^s)^x = g^{xs}$. Further, both client and node computes $H_2(g^{xs}, g^x, g^s, N, \mathsf{HybOR})$. For the lattice part the node computes $k_{1N} = p_C r_N + te'_N \approx ar_C r_N$ and the client computes $k_{1C} = p_N r_C + te_C \approx ar_C r_N$. The node also computes $\alpha = h^R(k_{1N})$ and sends it to the client. The client and node use α to make sure that the shared secret k_1 computed from k_{1N} (for the node) and k_{1C} (for the client) do not produce different results in modulo operation. They use the robust extractor $f^R(\cdot)$ (see Section 2.3) and compute $k_1 = f^R(k_{1N}, \alpha) = f^R(k_{1C}, \alpha)$. More details of the robust extractor can be found in [13]. After computing the shared secret k_1 the client and node both computes $H_1(k_1, p_C, p_N, N, \mathsf{HybOR})$. Further, from both parts of the shared secret they compute the session key and PRF key for the protocol as $(sk_m, sk) = H_1(k_1, p_C, p_N, N, \mathsf{HybOR}) \oplus H_2(g^{xs}, g^x, g^s, N, \mathsf{HybOR})$.

5 Security Analysis

5.1 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 be the partner with only 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 be partner with only 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.

5.2 Security against Type-I Adversary

We show that HybridOR is secure against Type-I adversary under either the GDH or the ring-LWE assumption. The motivation of this security theorem is to show that even if the ring-LWE assumption or the GDH assumption (but not both) is broken, HybridOR remains secure against Type-I adversary.

Theorem 6. *The protocol* HybridOR *is 1W-AKE secure against a PPT Type-I adversary under the GDH or the ring-LWE assumption, considering H_1 as randomness extractor and H_2 as random oracle. More precisely, for any PPT Type-I adversary A,*

$$Adv_A^{1w\text{-}ake} \leq \min(Adv^{drlwe}_{A \circ B_{0,1}} + Adv^{drlwe}_{A \circ B_{1,2}}, Adv^{GDH}_{A \circ B'}),$$

where $B_{0,1}$, $B_{1,2}$ and B' are the reduction algorithms as described in the proof.

Proof. To prove the security against a Type-I adversary, first we define a sequence of three games G_0 to G_2. Let E_i be the event that the adversary guesses bit b^* in game G_i.

G_0: This is the original 1W-AKE security game, where the reduction algorithm B generates all the public values honestly in all the sessions.

G_1: This game is identical to G_0, except here p_C is generated uniformly at random in the $test$ session.

G_2: This game is similar to G_1, except here p_N is generated uniformly at random in the $test$ session and also the $test$ session secret k_1 is generated uniformly at random.

As G_0 is the real 1W-AKE game, we can bound $\Pr(E_0)$ as

$$Adv_A^{1w\text{-}ake} = |\Pr(E_0) - 1/2|. \tag{1}$$

Lemma 7. *No PPT Type-I adversary can distinguish between G_0 and G_1 under the decision ring-LWE assumption, if H_1 is a randomness extractor and H_2 is a random oracle.*

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 efficiently 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 $p_C = ar_C + te_C$. In G_1, (a, p_C) are samples from a uniform distribution over $R_q \times R_q$. Under the decisional ring-LWE assumption these two distributions are indistinguishable.

Solving decision ring-LWE. To simulate the 1W-AKE challenger for A the reduction algorithm $B_{0,1}$ guesses ψ_i to be the $test$ session. In the $test$ session it honestly generates $(\mathbb{G}, g, g^x, g^s)$. $B_{0,1}$ 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^2. Thus if A can distinguish G_0 from G_1, $A \circ B_{0,1}$ can solve the decision ring-LWE problem. Hence,

$$|\Pr(E_0) - \Pr(E_1)| \leq Adv_{A \circ B_{0,1}}^{drlwe}. \tag{2}$$

□

Lemma 8. *No PPT Type-I adversary can distinguish between G_1 and G_2 under the decision ring-LWE assumption, if H_1 is a randomness extractor and H_2 is a random oracle.*

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 efficiently 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 $p_N = ar_N + te_N$. In G_2, (a, p_N) are samples from a uniform distribution over $R_q \times R_q$. Under the decisional ring-LWE assumption these two distributions are indistinguishable. In G_2, k_1 is also distributed as a random element from R_q. In both the cases p_C is uniformly distributed over R_q.

Solving decision ring-LWE. To simulate the 1W-AKE challenger for A the reduction algorithm $B_{1,2}$ guesses ψ_i to be the $test$ session. In the $test$ session it honestly generates $(\mathbb{G}, g, g^u, g^v)$. $B_{1,2}$ 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 $p_N = ar_N + te_N$ and $k_1 = p_C r_N + te'_N$. 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.

Thus if a PPT Type-I adversary A can distinguish between G_1 and G_2, then we can construct a reduction $B_{1,2}$ which can efficiently solve the ring-LWE problem. As a result we can write,

$$|\Pr(E_1) - \Pr(E_2)| \leq Adv^{drlwe}_{A \circ B_{1,2}}. \tag{3}$$

□

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 p_C, p_N and k_1 are chosen uniformly at random from 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

$$\Pr(E_2) = 1/2. \tag{4}$$

By combining equation $(1) - (4)$, we can write:

$$Adv^{1w\text{-}ake}_{A} \leq Adv^{drlwe}_{A \circ B_{0,1}} + Adv^{drlwe}_{A \circ B_{1,2}}. \tag{5}$$

Lemma 9. *The protocol* HybridOR *is 1W-AKE secure against a PPT Type-I adversary under the GDH assumption in the random oracle model.*

Proof. If there exists a PPT Type-I 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 $\{\mathbb{G}, g, g^u, g^v\}$ be the GDH challenge. B' has to compute g^{uv} in order to win the game.

The algorithm B' guesses ψ_i to be a $test$ session. To simulate the ψ_i, C runs the SetUp and generates (R, n, q, t, χ). It uses the group G and generator g from the GDH challenger in the public parameters. B' samples $a \leftarrow R$ sets (a, g^u) as the static key pair of the server and simulates ψ_i session by setting:

$$g^x = g^v,\ (p_C)_i = ar_C + te_C,\ (p_N)_i = ar_N + te_N,$$
$$(K_1)_i = (p_C)_i r_N + te'_N, (\alpha)_i = Signal((k_1)_i),$$

where, $r_C, r_N, e_C, e_N, e'_N \in_r \chi$. B' tosses a coin and chooses $b \in_r \{0, 1\}$. If $b = 0$ then B' computes the session key by computing $H_1((k_1)_i, (p_C)_i, (p_N)_i, N, \mathsf{HybOR}) \oplus H_2(\cdot, g^x, g^u, N, \mathsf{HybOR})$, 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.

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. 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 $H_2(Z, g^x, g^u, N, \mathsf{HybOR})$ for some $Z \in \mathbb{G}$, B' asks the DDH oracle whether (g^x, g^v, Z) is a valid DDH tuple. If that is the case, then $Z = g^{uv}$ 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. Hence we can write,

$$Adv^{1w\text{-}ake}_{A} \leq Adv^{GDH}_{A \circ B'}. \tag{6}$$

Note that for all other sessions with the same server, the reduction B' has to simulate the protocol correctly without the knowledge of the private key u. If not managed properly, simulation may fail due to inconsistent H_2 queries. B' uses the DDH oracle to respond consistently to the H_2 queries and the sk_reveal queries for the sessions that involve g^u. In particularl for H_2 queries that involve g^s, B' first verifies whether the shared secrets are computed honestly before responding with the session key. □

Conclusion. By combining equation (5) and (6), we prove the result.

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 other sessions. As a result the challenger can satisfy any kind of queries (see Section 3.2) from the adversary A during the simulation. □

5.3 Security against Type-II Adversary

In this section we show that in the pre-quantum computing era (today) HybridOR is secure against PPT Type-II adversary. Here we consider an active adversary which can actually become a partner with the ephemeral key of the server. We show that even in that case the adversary will not be able to win the 1W-AKE game.

Theorem 10. *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 $P = \{P_1, \cdots, P_m\}$ be the set of parties. Let $\{\mathbb{G}, g, g^u, g^v\}$ be the GDH challenge. B has to compute g^{uv} in order to win the GDH game.

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

$$
\begin{aligned}
&g^x = g^v,\\
&(p_C)_i = ar_C + te_C,\ (p_N)_i = ar_N + te_N,\\
&(K_1)_i = (p_C)_i r_N + te'_N, (\alpha)_i = Signal((k_1)_i),
\end{aligned}
$$

where, $r_C, r_N, e_C, e_N, e'_N \in_r \chi$. B tosses a coin and chooses $b \in_r \{0, 1\}$. If $b = 0$ then B computes the session key by computing $H_1((k_1)_i, (p_C)_i, (p_N)_i, N, \mathsf{HybOR}) \oplus H_2(\cdot, g^x, g^u, N, \mathsf{HybOR})$, 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.

The adversary 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 A with the correct value of r_N. A can compute $H_1((k_1)_i, (p_C)_i, (p_N)_i, N,$ HybOR) using that information. 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. 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 $H_2(Z, g^x, g^u, N,$ HybOR) for some $Z \in \mathbb{G}$, B asks the DDH oracle whether (g^x, g^v, Z) is a valid DDH tuple. If that is the case, then $Z = g^{uv}$ 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 solve the GDH problem with significant probability.

Note that for all other sessions with the same server, the reduction B has to simulate the protocol correctly without the knowledge of the private key u. If not managed properly, simulation may fail due to inconsistent H_2 queries. B uses the DDH oracle to respond consistently to the H_2 queries and the sk_reveal queries for the sessions that involve g^u. In particular for H_2 queries that involve g^s, B first verifies whether the shared secrets are computed honestly before responding with the session key. □

Notice that this theorem directly imply that a Type-II adversary can break the 1W-AKE security of HybridOR in a quantum world. As in the quantum world the Type-II adversary can compute the discrete log of the long term secret g^s and it is already partner to the ephemeral secret p_N in the $test$ session. Hence the adversary can compute the session key and wins the 1W-AKE game.

In order to make this protocol secure against Type-II adversary in a quantum world we need replace the long term key with a quantum secure component. But in that case we cannot use our current DH public keys and subsequently requires modification to the current Tor public key infrastructure. So, in today's scenario it is sufficient to follow the HybridOR design in the current form. As we can deploy this easily in the current Tor network.

5.4 Security against Type-III Adversary

A more important question to ask now is whether HybridOR provides forward security in the post-quantum world. If not, then the privacy of today's anonymous communication cannot be ensured once quantum computers arrive. We prove that HybridOR is forward secure if the ring-LWE problem is hard. The motive of this theorem is to show that by using HybridOR in Tor we can aim at the privacy of today's anonymous communication even after quantum computers arrive.

Theorem 11. HybridOR *is 1W-AKE secure against a PPT Type-III adversary under the ring-LWE assumption. More precisely, for any PPT Type-III adversary A,*

$$Adv_A^{\textit{1w-ake}} \leq Adv_{A \circ B_{0,1}}^{\textit{drlwe}} + Adv_{A \circ B_{1,2}}^{\textit{drlwe}},$$

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 in all the sessions.

G_1: This game is identical to G_0, except here p_C is generated uniformly at random in the $test$ session.

G_2: This game is similar to G_1, except here p_N is generated uniformly at random in the $test$ and also the $test$ session secret k_1 is generated uniformly at random.

Lemma 12. *No PPT Type-III adversary can distinguish between G_0 and G_1 under the decision ring-LWE assumption, if H_1 is a randomness extractor.*

Lemma 13. *No PPT Type-III adversary can distinguish between G_1 and G_2 under the decision ring-LWE assumption, if H_1 is a randomness extractor.*

To prove Lemma 12 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 13, 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 11 is same as the proof of Theorem 6 (see Lemma 7, Lemma 8). □

Quantum safe reduction. In [46] Song pointed out that a post-quantum secure scheme against a classical adversary does not immediately guarantee that the scheme is also secure against a quantum adversary. Song gives conditions under which a classical proof can be lifted to provide quantum security. One of the condition is that the classical reduction is a *straight-line* reduction. That means that the reduction runs the adversary from start to end without any rewinding or restarting. Our reductions against Type-III adversary are straight-line, hence they satisfy Song's criterion for security against a quantum adversary.

6 Performance Analysis

We analyze the performance of HybridOR, and compare it with the ntor protocol.

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 [34]. Similar to other ring-LWE based protocols [6, 40, 48], the security and performance of HybridOR essentially depend on the three factors: n, q, and β. 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 σ of the error distribution χ.

Lindner and Peikert [32] show how the parameters (n, q, β) 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 [20, 45] 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 R_q.

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

Computation Cost. We assume that the elements r_C, e_C, p_C and g^x are precomputed 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 R_q and 1 exponentiation in $\mathbb{G}$.

Multiplications over R_q can be performed efficiently using an FFT-based algorithm [34], which takes $O(n \log n)$ for a serial implementation and $O(\log n)$ time for a parallel implementation [25]. 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 [26] 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 R_q only requires $\approx 50\mu s$, and the exponentiation in $\mathbb{G}$ requires $\approx 100\mu s$.

The ntor 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 unoptimized proof-of-concept HybridOR implementation is nearly 1.5 times faster than the ntor protocol used in Tor. 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.

Communication Cost. In the HybridOR protocol the client has to send an element $g^x \in \mathbb{G}$ and an element $p_C \in R_q$ to the node. We require 32 bytes to represent an element on Curve25519. On the other hand, for an element in R_q, we require at most $1/8(n \lg q)$ bytes, which is around 1280 bytes for the chosen parameter set $(512, 1051649, 8.0)$. Therefore, the client communicates 1312 bytes to the server.

On the other hand, the node has to send an element $p_N \in R_q$, an n-bit signal α, and the key confirmation message of 32 bytes to the client. That requires a total of $1/8(n \lg q + n) + 32$ bytes. For the chosen parameter set $(512, 1051649, 8.0)$, the node has to send about 1376 bytes to the client.

The current Tor implementation employs 512-byte cells; thus, for HybridOR, the client and the node each will have to communicate three cells. In comparison, for the currently employed ntor protocol, a single cell from the client and the server suffices. However, it is possible to use smaller value for q without affecting the security, which can reduce the communication overhead of the protocol.

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 based 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 arguably 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.

Acknowledgements. We would like to thank Sujoy Sinha Roy for his suggestions regarding the ring-LWE implementation. This work was supported by the German Universities Excellence Initiative, and was partially supported by CoEC and R.C. Bose Centre for Cryptology and Security, ISI Kolkata.

References

[1] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *Advances in Cryptology—CRYPTO '09*, pages 595–618. 2009.

[2] M. Backes, I. Goldberg, A. Kate, and E. Mohammadi. Provably Secure and Practical Onion Routing. In *Proc. 25th IEEE Computer Security Foundations Symposium (CSF)*, 2012.

[3] M. Backes, A. Kate, and E. Mohammadi. Ace: An Efficient Key-Exchange Protocol for Onion Routing. In *WPES*, pages 55–64. ACM, 2013.

[4] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology—CRYPTO '93*, pages 232–249. 1994.

[5] D. J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In *PKC '06*, pages 207–228, 2006.

[6] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. *IACR Cryptology ePrint Archive*, 2014, 2014. To appear at IEEE Security and Privacy Symposium '15.

[7] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé. Classical Hardness of Learning with Errors. In *STOC '13*, pages 575–584, 2013.

[8] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS '01*, pages 136–145, 2001.

[9] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology—EUROCRYPT 2001*, pages 453–474. 2001.

[10] D. Catalano, D. Fiore, and R. Gennaro. Certificateless Onion Routing. In *Proc. 16th ACM Conference on Computer and Communication Security (CCS)*, pages 151–160, 2009.

[11] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 4(2):84–88, 1981.

[12] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proc. 24th IEEE Symposium on Security & Privacy*, pages 2–15, 2003.

[13] J. Ding, X. Xie, and X. Lin. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. Cryptology ePrint Archive, Report 2012/688, 2012. `http://eprint.iacr.org/`.

[14] R. Dingledine and N. Mathewson. Design of a Blocking-Resistant Anonymity System. Technical report. `https://svn.torproject.org/svn/projects/design-paper/blocking.pdf`.

[15] R. Dingledine and N. Mathewson. Tor Protocol Specification. `https://gitweb.torproject.org/torspec.git?a=blob_plain;f=tor-spec.txt`.

[16] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium (USENIX)*, pages 303–320, 2004.

[17] L. Ducas and A. Durmus. Ring-LWE in Polynomial Rings. In *PKC '12*, pages 34–51, 2012.

[18] P.-A. Fouque, D. Pointcheval, and S. Zimmer. HMAC is a Randomness Extractor and Applications to TLS. In *ACM ASIACCS '08*, pages 21–32, 2008.

[19] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Strongly Secure Authenticated Key Exchange from Factoring, Codes, and Lattices. In *PKC '12*, pages 467–484, 2012.

[20] A. Fujioka, K. Suzuki, K. Xagawa, and K. Yoneyama. Practical and Post-Quantum Authenticated Key Exchange from One-Way Secure Key Encapsulation Mechanism. In *ACM ASIACCS '13*, pages 83–94, 2013.

[21] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC '09*, pages 169–178, 2009.

[22] I. Goldberg. On the Security of the Tor Authentication Protocol. In *Proc. 6th Workshop on Privacy Enhancing Technologies*, pages 316–331, 2006.

[23] I. Goldberg. Privacy Enhancing Technologies for the Internet III: Ten Years Later. In *Digital Privacy: Theory, Technologies and Practices*, pages 3–18, 2007.

[24] I. Goldberg, D. Stebila, and B. Ustaoglu. Anonymity and One-Way Authentication in Key Exchange Protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013.

[25] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss. On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. In *CHES 2012*, pages 512–529. 2012.

[26] T. Granlund and the GMP development team. *GMP: The GNU Multiple Precision Arithmetic Library*, 6.0 edition, 2014. `http://gmplib.org/`.

[27] A. Kate and I. Goldberg. Using Sphinx to Improve Onion Routing Circuit Construction. In *FC '10*, pages 359–366, 2010.

[28] A. Kate, G. M. Zaverucha, and I. Goldberg. Pairing-Based Onion Routing. In *In PETS '07*, pages 95–112, 2007.

[29] A. Kate, G. M. Zaverucha, and I. Goldberg. Pairing-Based Onion Routing with Improved Forward Secrecy. *ACM Trans. Inf. Syst. Secur.*, 13(4):29, 2010.

[30] H. Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. In *Advances in Cryptology—CRYPTO 2005*, pages 546–566. 2005.

[31] H. Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF scheme. In *Advances in Cryptology—CRYPTO '10*, volume 6223, pages 631–648. 2010.

[32] R. Lindner and C. Peikert. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *CT-RSA 2011*, pages 319–339. 2011.

[33] V. Lyubashevsky and D. Micciancio. Asymptotically Efficient Lattice-Based Digital Signatures. In *Theory of Cryptography Conference (TCC) '08*, pages 37–54. 2008.

[34] V. Lyubashevsky, C. Peikert, and O. Regev. On Ideal Lattices and Learning with Errors over Rings. *J. ACM*, 60(6):43:1–43:35, Nov. 2013.

[35] D. Micciancio. Improving Lattice Based Cryptosystems Using the Hermite Normal Form. In *Cryptography and Lattices*, volume 2146 of *LNCS*, pages 126–145. 2001.

[36] D. Micciancio and O. Regev. Lattice-based Cryptography. In *Post-Quantum Cryptography*, pages 147–191. 2009.

[37] T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. In *PKC '01*, pages 104–118, 2001.

[38] L. Øverlier and P. Syverson. Improving Efficiency and Simplicity of Tor Circuit Establishment and Hidden Services. In *In PETS '07*, pages 134–152, 2007.

[39] C. Peikert. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem: Extended Abstract. In *STOC '09*, pages 333–342, 2009.

[40] C. Peikert. Lattice Cryptography for the Internet. Cryptology ePrint Archive, Report 2014/070, 2014. `http://eprint.iacr.org/`.

[41] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE J-SAC*, 16(4):482–494, 1998.

[42] O. Regev. New Lattice-Based Cryptographic Constructions. *J. ACM*, 51(6):899–942, Nov. 2004.

[43] O. Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *STOC '05*, pages 84–93, 2005.

[44] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *ACM WPES '02*, pages 91–102, 2002.

[45] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede. Compact Ring-LWE based Cryptoprocessor. Cryptology ePrint Archive, Report 2013/866, 2013. `http://eprint.iacr.org/`.

[46] F. Song. A note on quantum security for post-quantum cryptography. In *Post-Quantum Cryptography*, pages 246–265. 2014.

[47] The Tor Project. `https://www.torproject.org/`, 2003. Accessed November 2014.

[48] J. Zhang, Z. Zhang, J. Ding, M. Snook, and O. Dagdelen. Authenticated Key Exchange from Ideal Lattices. Cryptology ePrint Archive, Report 2014/589, 2014. `http://eprint.iacr.org/` To appear at Eurocrypt '15.

A Comparison Between the ntor, Ace, and HybridOR Protocols

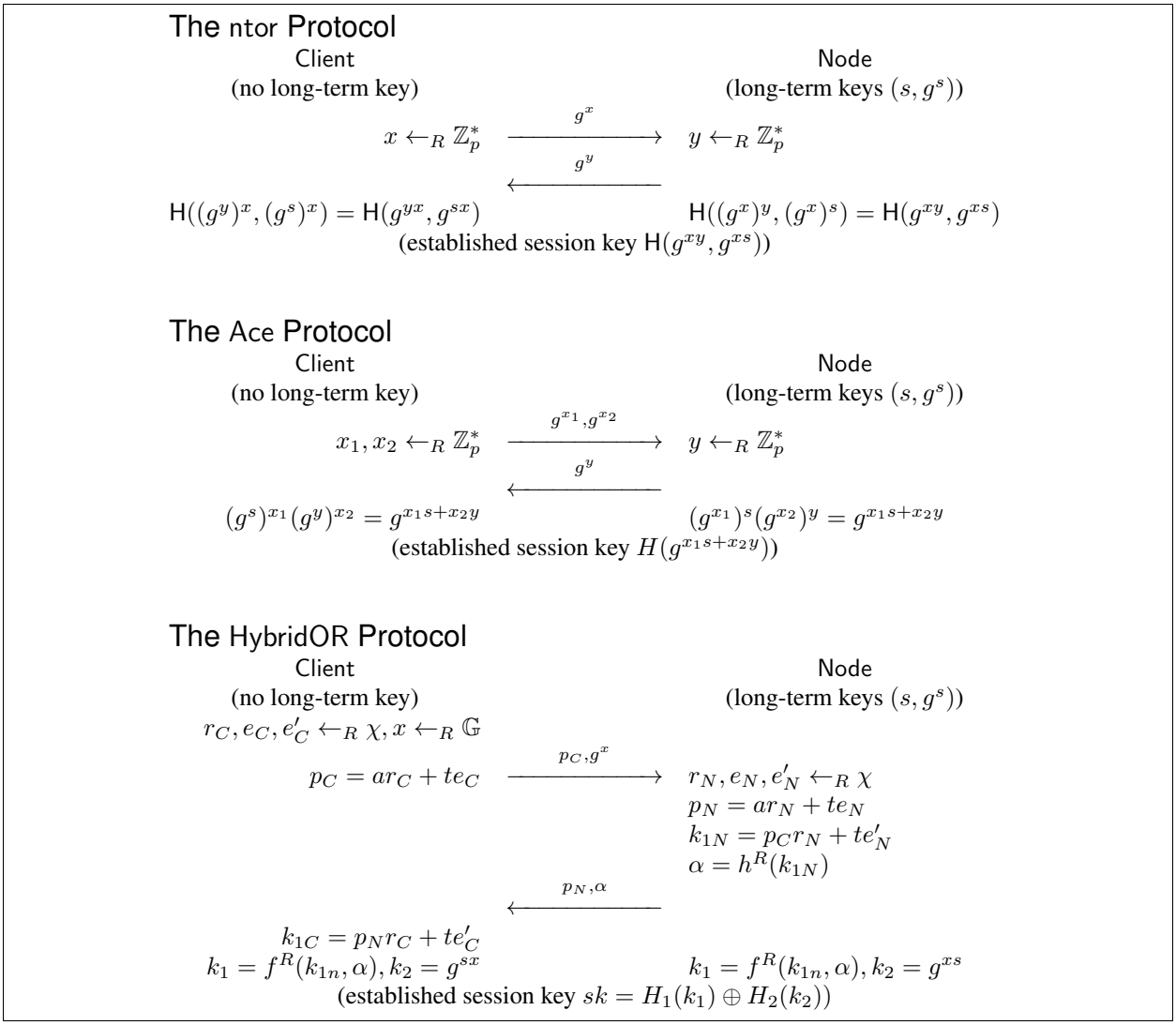

Figure 4: A comparative overview of the ntor, Ace, and HybridOR protocols: For readability, we neglect the information used for the key derivation and confirmation messages.