John M. Schanck*, William Whyte, and Zhenfei Zhang

Circuit-extension handshakes for Tor achieving forward secrecy in a quantum world

Abstract: We propose a circuit extension handshake for Tor that is forward secure against adversaries who gain quantum computing capabilities after session negotiation. In doing so, we refine the notion of an authenticated and confidential channel establishment (ACCE) protocol and define pre-quantum, transitional, and post-quantum ACCE security. These new definitions reflect the types of adversaries that a protocol might be designed to resist. We prove that, with some small modifications, the currently deployed Tor circuit extension handshake, ntor, provides pre-quantum ACCE security. We then prove that our new protocol, when instantiated with a post-quantum key encapsulation mechanism, achieves the stronger notion of transitional ACCE security. Finally, we instantiate our protocol with NTRUEncrypt and provide a performance comparison between ntor, our proposal, and the recent design of Ghosh and Kate.

DOI 10.1515/popets-2016-0037
Received 2016-02-29; revised 2016-06-02; accepted 2016-06-02.

1 Introduction

A key exchange protocol allows two parties who share no common secrets to agree on a common key over a public channel. In addition to achieving this basic goal, key exchange protocols may satisfy various secondary properties that are deemed important to security in a particular setting. Modern key exchange protocols typically provide one or more of the following features.

1. **Authentication:** If one or both parties can be assured of their peer's identity, the key exchange is *authenticated*. The authentication is *one-way* in the first case and *mutual* in the latter.
2. **Anonymity:** Some one-way authenticated key exchange protocols, such as ntor [13], guarantee that the unauthenticated peer does not reveal their identity just by participating in the protocol. Such protocols are deemed *one-way anonymous*.
3. **Forward Secrecy:** A protocol provides forward secrecy if the compromise of a party's long-term key material does not affect the secrecy of session keys negotiated prior to the compromise. Forward secrecy is typically achieved by mixing long-term key material with ephemeral keys that are discarded as soon as the session has been established.

Forward secret protocols are a particularly effective tool for resisting mass surveillance as they resist a broad class of *harvest-then-decrypt* attacks. In a harvest-then-decrypt attack a passive adversary records ciphertexts in the present with the hope of acquiring new decryption capabilities in the future. Without forward secrecy any number of non-cryptanalytic attacks may lead to a loss of confidentiality. For example, a server's long-term key may be compromised by a hacker or subpoenaed by a court. With forward secrecy, the only way that a passive adversary can learn a session key that was negotiated pre-compromise is through cryptanalysis. For some applications, this makes forward secrecy an essential requirement.

Unfortunately, all of the key exchange protocols in widespread deployment are vulnerable to quantum cryptanalysis. A reasonable case could therefore be made that the use of *post-quantum* primitives is a prerequisite for forward secrecy.

4. **Post-quantum security:** A cryptographic primitive or protocol is deemed *post-quantum* if it is secure against adversaries that can perform polynomial time quantum computations.

Primitives based on the hardness of discrete logarithms (Diffie-Hellman, ECDH, DSA, ECDSA) and integer factorization (RSA) can be broken in quantum polynomial time using Fourier sampling techniques [6, 29]. There are, however, a number of primitives that are believed to resist quantum adversaries, and these could be used to construct post-quantum, forward secure, authenticated key exchange.

Recent announcements by NSA [10] and NIST [9] have made it clear that government users are seriously considering migrating to post-quantum cryptography in

***Corresponding Author: John M. Schanck:** University of Waterloo and Security Innovation, email: jschanck@securityinnovation.com
William Whyte: Security Innovation, email: wwhyte@securityinnovation.com
Zhenfei Zhang: Security Innovation, email: zzhang@securityinnovation.com

the near future. The exact migration path remains unclear, but several factors may slow the transition to post-quantum algorithms, e.g.

1. low confidence in the security of new primitives,
2. low confidence in the reliability of new implementations,
3. the need for compliance with existing standards, and
4. difficulty integrating new primitives into existing protocols and public key infrastructures.

The first three issues may be addressed by "hybridizing" well-established and post-quantum systems. For example, recent work on integrating Ring-LWE into TLS considered a ciphersuite that performs a Ring-LWE key exchange in parallel with an ECDH key exchange [7]. If an attack is found on the Ring-LWE parameter set, or its implementation, the proposed ciphersuite maintains all of the security that would have been provided by ECDH on its own.

The fourth issue may be addressed, partially, by upgrading to post-quantum primitives only where it is absolutely necessary. The TLS ciphersuites of [7], as well as the Tor circuit extension handshakes of [12] and the present work, forego post-quantum authentication mechanisms because they deem such mechanisms to be unnecessary in the short term. These protocols cannot defend against adversaries with quantum capabilities at the time of session negotiation. They are, however, secure against adversaries who gain quantum computing capabilities sometime after session negotiation. This is a useful class of adversaries to defend against, and we believe that this class accurately represents adversaries in the real world today.

We will say that a protocol provides *transitional security* if it provides pre-quantum authentication and post-quantum confidentiality. Such protocols are safe to use in the current transitional period between the pre- and post-quantum settings.

Pre-quantum, transitional, and post-quantum security must be defined with respect to a specific security experiment in order to be meaningful. The exact sense in which we will use these terms is provided by Definitions 2.7, 2.8, and 2.9.

1.1 Our contribution

The ntor protocol [13] is a forward secret, one-way authenticated key exchange protocol that has been deployed in Tor since version 0.2.4.8-alpha [23]. It relies on ephemeral ECDH keys for forward secrecy and, consequently, is vulnerable to harvest-and-decrypt attacks involving quantum adversaries.

We show how to incorporate a secondary key encapsulation mechanism (KEM) into the ntor protocol to strengthen its resistance to harvest-and-decrypt attacks. We describe a modular protocol, hybrid, in Section 3 that allows for the incorporation of zero or more KEMs.

Inspired by recent work on the provable security of TLS and SSH ciphersuites as-standardized [2, 18, 21], we give proofs in the authenticated and confidential channel establishment (ACCE) model. The ACCE model was introduced in [18]; we review the ACCE model in Section 2 before refining it with notions of pre-quantum, transitional, and post-quantum ACCE security (Definitions 2.7, 2.8, and 2.9). These ACCE variants reflect the types of adversaries that a protocol is designed to resist. A pre-quantum ACCE protocol provides pre-quantum authentication and pre-quantum confidentiality. A transitional ACCE protocol provides pre-quantum authentication and post-quantum confidentiality. A post-quantum ACCE protocol provides post-quantum authentication and post-quantum confidentiality.

We prove that hybrid, with zero additional KEMs, is a pre-quantum ACCE protocol. With zero KEMs hybrid is essentially the same as ntor. We describe the differences in Section 3.2; the modifications allow us to prove the security of hybrid under slightly weaker assumptions than were employed in the proof of security for ntor in [13].

We then prove that hybrid with one post-quantum KEM is a transitional ACCE protocol. More specifically, we show that an adversary's advantage in violating the channel security of hybrid is a function of its minimal advantage against either the Diffie-Hellman primitive or the secondary KEM.

Finally, we provide a concrete instantiation of our protocol with a single additional NTRUEncrypt-based KEM. Based on the conjectured post-quantum security of NTRUEncrypt we claim that this instantiation is transitionally secure. We provide an implementation and performance figures for this instantiation, and compare it with recent work [12] based on Ring-LWE.

We summarize the performance comparison between the legacy Tor handshake (tap), the current Tor handshake (ntor), our protocol (hybrid), and the proposal of Ghosh-Kate in Table 1.1. We discuss the performance results in Section 5.

We make this proposal for two reasons. First, we believe it to be an interesting case study into the prac-

ticality of post-quantum cryptography and into the difficulties one might encounter when transitioning to post-quantum primitives within real-world protocols and code-bases. Second, we believe that Tor is a strong candidate for an early transition to post-quantum primitives. Users of Tor may be justifiably concerned about adversaries who record traffic in the present and store it for decryption when technology or cryptanalytic techniques improve.

1.2 Related work

The ntor protocol was analyzed in a variant of the extended Canetti-Krawczyk (eCK) model with support for one-way authentication [13]. The proof of security relies on the Gap Diffie-Hellman assumption and makes extensive use of random oracles.

Ghosh and Kate [12] propose a transitionally secure circuit extension handshake for Tor that relies on Ring-LWE for its forward secrecy against post-quantum adversaries. Their proof is in the same one-way authenticated key exchange model used in [13], and makes similar assumptions. There are a number of significant differences between their work and ours beyond the security model. In particular their protocol does not "fall back" to ntor in the event that Ring-LWE is found to be completely insecure. Specifically, if Ring-LWE is found to be insecure against pre-quantum adversaries, then the protocol of [12] fails to provide forward secrecy against pre-quantum adversaries. The protocols also differ significantly in their key derivation methods. On the positive side, their protocol is quite likely to be faster than ours as it performs fewer Diffie-Hellman operations.

Bos, Costello, Naehrig, and Stebila have proposed transitionally secure ciphersuites for TLS based on signed Ring-LWE [7]. The same work proposes hybrid ciphersuites that incorporate both elliptic curve Diffie-Hellman and Ring-LWE shares. The security of the signed Ring-LWE ciphersuites is proven in the ACCE framework, however no proof is given for the hybrid ciphersuites.

1.3 Notation

We distinguish objects by typeface: algorithms $\mathcal{A}$ and oracles $\mathcal{O}$, Primitives and protocols, sets $\mathbb{E}$, ordered sets $\vec{\mathbb{E}}$, groups $\mathbf{G}$, and `strings`. Sampling uniformly from a set is denoted $x \xleftarrow{\$} \mathbb{S}$. Assignment from a function $f(\cdot)$, even if f is randomized, is denoted $x = f(y)$. Persistent program state will be denoted by Greek letters, in particular π. Assignment to a persistent variable within a given persistent state π is written $\pi.x = y$. We will access persistent state by writing x rather than $\pi.x$ provided that doing so is completely unambiguous.

2 Security model

The *authenticated and confidential channel establishment* (ACCE) model was proposed by Jager, Kohlar, Schäge, and Schwenk to prove the security of TLS with signed ephemeral Diffie-Hellman ciphersuites and mutual authentication [18]. The model has been successfully extended to yield proofs of security for a variety of real-world protocols. Extensions to more common setting of one-way, *server-only*, authentication were provided by Kohlar, Schäge, and Schwenk [19], and by Krawczyk, Paterson, and Wee [21]. The server-only ACCE model was later used by Bos, Costello, Naehrig, and Stebila to prove the security of Ring-LWE based TLS ciphersuites [7].

2.1 Security definitions

An ACCE protocol is a two-party secure communition protocol consisting of two phases. In the first phase, the *pre-accept* phase, the parties exchange a key. In the *post-accept* phase the parties use the key to exchange messages encrypted with an authenticated encryption scheme.

The ACCE execution environment allows us to model the concurrent execution of one or more protocols by multiple parties. The environment involves n_P parties, $\mathbb{P} = \{P_i : i \in \{1, \ldots, n_P\}\}$, each of whom may be involved in at most n_S sessions. The s-th session involving party i is modeled by a stateful *session oracle* π_i^s. We refer to both the oracle and its collection of internal state as π_i^s. The internal state kept by these oracles is as follows:

		tap	ntor	hybrid	Ghosh-Kate
data	client → server bytes	186	84	693	1312
	server → client bytes	148	64	673	1376
computation	client init	$258\mu s$	$84\mu s$	$661\mu s$	$150\mu s^*$
	server response	$682\mu s^\dagger$	$263\mu s$	$306\mu s$	$150\mu s^*$
	client finish	$233\mu s$	$180\mu s$	$218\mu s$	$150\mu s^*$
	total	$1173\mu s$	$527\mu s$	$1185\mu s$	$450\mu s^*$
	% client	42%	50%	74%	67%

Table 1.1. Performance comparison of tap, ntor, hybrid, and Ghosh-Kate. The hybrid protocol was instantiated as in Section 5 with ntruees443ep1.

† The tap benchmark reports two cases, this is the "guessed right" case. The other value was 890μs.

$*$ Estimates from [12]. Assumes $100\mu s$ per Diffie-Hellman group operation and $50\mu s$ for one multiplication and one addition in the R-LWE ring. All other costs, such as sampling the R-LWE secrets, are ignored.

Definition 2.1 (Per-session variables [2]). Let π_i^s denote the following collection of *per-session* variables:

- $\rho \in \{\texttt{init}, \texttt{resp}\}$: The party's role in this session.
- $pid \in \{1, \ldots, n_P, \perp\}$: The identifier of the alleged peer for this session, or $\perp$ for an unauthenticated peer.
- $\alpha \in \{\texttt{in-progress}, \texttt{reject}, \texttt{accept}\}$: The status.
- K: A session key, or $\perp$ if a key has not been negotiated. Note that K may consist of several concatenated sub-keys.
- sid: A session identifier defined by the protocol.
- st_e, st_d: State for the stateful authenticated encryption and decryption algorithms.
- Any additional state specific to the protocol.
- Any additional state specific to the security experiment.

The state of all $n_P n_S$ session oracles is initially uninitialized with the exception of a random bit, $\pi_i^s.b \xleftarrow{\$} \{0,1\}$, that is required by the security experiment.

All interaction between the session oracles is mediated by an oracle, $\mathcal{C}$, called the challenger. The challenger has a global view of the execution environment and may manipulate the state of session oracles. The following queries to $\mathcal{C}$ are allowed.

- $\mathsf{Send}(\pi_i^s, m) \to m'$. Causes the oracle π_i^s to execute the next routine of its handshake protocol with input m. The input m may be a special initialization string that specifies the protocol, the role, and (optionally) a peer identifier. In this case π_i^s sets $\pi_i^s.\rho$ and $\pi_i^s.pid$ accordingly. Otherwise π_i^s processes the message in accordance with the protocol specification, updates its internal state, and (optionally) outputs an outgoing message m'. Note that if π_i^s has reached the `accept` or `reject` state then it will return $\perp$ to any Send query issued.
- $\mathsf{Reveal}(\pi_i^s) \to \pi_i^s.K$. Returns the session key $\pi_i^s.K$ if it is initialized, otherwise $\perp$. Note that $\pi_i^s.K$ is initialized iff $\pi_i^s.\alpha = \texttt{accept}$.
- $\mathsf{Corrupt}(P_i) \to sk$. Returns the long-term secret of party P_i. This may allow the adversary to simulate P_i by forging Send queries.
- $\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \texttt{len}, H) \to C$ or $\perp$. If the session key $\pi_i^s.K$ is initialized then this query causes the oracle π_i^s to encrypt a message under its session key. Otherwise the oracle returns $\perp$. The exact behavior may be found in Algorithm 2.1, but this depends on the specifics of the security experiment (Section 2.2) and the definition of stateful length-hiding authenticated encryption (Section 2.7).
- $\mathsf{Decrypt}(\pi_i^s, C, H) \to m$ or $\perp$. If the $\pi_i^s.K$ is initialized then this query causes the oracle π_i^s to execute Algorithm 2.2 and return its output. Otherwise the oracle returns $\perp$.

Remark 2.2. Queries to distinct oracles may be issued in parallel, but a partial order is kept to properly handle Corrupt queries. More formally, we say P_j is τ_j-corrupted if the τ_j-th query was $\mathsf{Corrupt}(P_j)$.

2.2 Server-only ACCE security

We follow [21] in our definition of server-only ACCE (SACCE) protocols.

The server-only ACCE (SACCE) security of a protocol is defined by an experiment involving the challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The challenger simulates n_P parties with whom the adversary interacts via Send, Reveal, Corrupt, Encrypt, and Decrypt queries. We make no restrictions on the computational power of the adversary. The adversary may, for example, per-

form quantum computation. That said, we require all communication with the session oracles to be classical.

The parties are partitioned into servers $\mathbb{S}$ and clients $\mathbb{C}$; $\mathbb{P} = \mathbb{S} \cup \mathbb{C}$. Servers are permitted to take the initiator role (i.e. to act as clients), but clients cannot take the responder role.

At the beginning of the game the challenger generates long-term public/private key pairs for all servers and reveals the public keys to the adversary. The adversary submits any number of Send, Reveal, Corrupt, Encrypt, and Decrypt queries, then ends the experiment by outputting a triple

$$(i, s, b') \in \{1, \ldots, n_P\} \times \{1, \ldots, n_S\} \times \{0, 1\}.$$

The adversary is successful if it violates either the authenticity of the key exchange, or the security of the established channel. The following definitions formalize the adversary's success criteria.

Definition 2.3 (Matching Sessions). We say that π_i^s matches π_j^t if

- $\pi_i^s.\rho \neq \pi_j^t.\rho$ (the roles are distinct)
- $\pi_i^s.sid = \pi_j^t.sid$ (the sessions have identical transcripts).

Intuitively, matching sessions that have reached the accept state have successfully negotiated an authentic key. An adversary violates the authenticity of the key exchange if it causes a party to accept without a matching session. The following definition captures the exact conditions in which an adversary succeeds in violating the authenticity of a protocol. Note that this definition is specific to one-way authenticated protocols in which the authenticated party is the responder.

Definition 2.4 (π_i^s accepts maliciously). Let π_i^s be a session. We say that π_i^s *accepts maliciously* if

- $\pi_i^s.\rho = \texttt{init}$;
- $\pi_i^s.\alpha = \texttt{accept}$;
- $\pi_i^s.pid = j \neq \perp$ and no Corrupt(j) query was issued before π_i^s accepted; and
- there is no unique session π_j^t that matches π_i^s.

For a one-way authenticated protocol Π and an adversary $\mathcal{A}$, $\mathrm{Adv}_{\Pi}^{\texttt{sacce-sa}}(\mathcal{A})$ is the probability that there exists a session that has accepted maliciously when $\mathcal{A}$ terminates.

Definition 2.5 (Channel Security). We say that $\mathcal{A}$ *answers the encryption challenge correctly* if, when $\mathcal{A}$ terminates with output (i, s, b'),

- $\pi_i^s.\alpha = \texttt{accept}$;
- $\pi_i^s.pid = j \neq \perp$;
- π_i^s did not accept maliciously;
- π_i^s accepted in response to query τ_0 and party P_j is τ_j-corrupted with $\tau_0 < \tau_j$;
- $\mathcal{A}$ did not issue Reveal(π_i^s) nor Reveal(π_j^t) for any (j, t) such that π_j^t matches π_i^s;
- $\pi_i^s.b = b'$.

For a protocol Π, $\mathrm{Adv}_{\Pi}^{\texttt{sacce-ae}}(\mathcal{A}) = |p - 1/2|$ where p is the probability that $\mathcal{A}$ answers the encryption challenge correctly.

These definitions naturally correspond to security experiments that we denote `sacce-sa` (SACCE Server Authentication) and `sacce-ae` (SACCE Authenticated Encryption) respectively.

Remark 2.6. Our definition of channel security differs from that given in previous work such as [2, 18, 21]. Specifically we have added the requirement that π_i^s has not accepted maliciously. This change allows us to distinguish between adversaries that can *only* violate channel security by violating authenticity, and those that can violate channel security without active intervention in the pre-accept stages. In turn this allows us to model security against harvest-then-decrypt attacks where the adversary gains quantum capabilities after the session has been negotiated.

Definition 2.7 (Pre-quantum SACCE Security).
A protocol Π provides *pre-quantum SACCE security* if the quantity

$$\mathrm{Adv}_{\Pi}^{\texttt{sacce-sa}}(\mathcal{A}) + \mathrm{Adv}_{\Pi}^{\texttt{sacce-ae}}(\mathcal{A})$$

is a negligible function of the security parameter λ for all PPT pre-quantum adversaries $\mathcal{A}$.

Definition 2.8 (Transitional SACCE Security).
A protocol Π provides *transitional SACCE security* if the quantity

$$\mathrm{Adv}_{\Pi}^{\texttt{sacce-sa}}(\mathcal{A}) + \mathrm{Adv}_{\Pi}^{\texttt{sacce-ae}}(\mathcal{Q})$$

is a negligible function of the security parameter λ for all PPT pre-quantum adversaries $\mathcal{A}$ and PPT post-quantum adversaries $\mathcal{Q}$.

Definition 2.9 (Post-quantum SACCE Security).
A protocol Π provides *post-quantum SACCE security* if the quantity

$$\mathrm{Adv}_{\Pi}^{\texttt{sacce-sa}}(\mathcal{Q}) + \mathrm{Adv}_{\Pi}^{\texttt{sacce-ae}}(\mathcal{Q})$$

is a negligible function of the security parameter λ for all PPT post-quantum adversaries $\mathcal{Q}$.

Remark 2.10. The restriction on malicious acceptance in Definition 2.5 is necessary to distinguish between transitional and post-quantum ACCE protocols. Without it there are no transitional protocols that are not fully post-quantum. The restriction has no effect on the classes of pre- and post-quantum ACCE protocols; a pre- or post-quantum ACCE protocol that is secure using our definition would also be secure if malicious acceptance were allowed in Definition 2.5. The converse is true as well.

2.3 Selective SACCE security

In order to simplify our proofs, we will make use of the *selective* variants of the authentication and channel security experiments introduced in [21]. We denote these by s-sacce-sa and s-sacce-ae respectively. In the selective authentication experiment the adversary must commit to the index (i^*, s^*) of a session that will accept maliciously. In the selective channel security experiment the adversary must commit to indices for a pair of matching sessions, i.e. (i^*, s^*) and (j^*, t^*).

Lemma 2.11 (Adapted from [21]). *For any adversary $\mathcal{A}$ (pre-quantum or post-quantum), there exists an adversary $\mathcal{B}$ such that*

$$\mathrm{Adv}^{\text{sacce-sa}}_{\Pi}(\mathcal{A}) \leq n_S n_P \,\mathrm{Adv}^{\text{s-sacce-sa}}_{\Pi}(\mathcal{B})$$
$$\mathrm{Adv}^{\text{sacce-ae}}_{\Pi}(\mathcal{A}) \leq n_S^2 n_P \,\mathrm{Adv}^{\text{s-sacce-ae}}_{\Pi}(\mathcal{B})$$

Note that the corresponding lemma given in [21] has an additional factor of $n_S \,\mathrm{Adv}^{\text{s-sacce-sa}}_{\Pi}(\mathcal{B})$ in the reduction from sacce-ae to s-sacce-ae. This is because that work allows sessions to have accepted maliciously in the channel security game.

2.4 Summary of changes to ACCE model

- We explicitly allow adversaries to perform quantum computations, but we require that communication between the adversary and challenger is classical. We justify our decision not to use a fully quantum model in Section 6.
- Our definition of channel security differs from previous work, such as [18, 21], in that we require that the session indicated by the adversary has not accepted maliciously.

2.5 Assumptions on primitives

We will need two length parameters: μ and λ. We will assume that $\mu = 2\lambda$ and that λ is the intended bit-security of the handshake. All cryptographic primitives are expected to provide at least λ-bit pre-quantum security; symmetric primitives and KEMs should provide λ-bit post-quantum security. This implies that the length of a Diffie-Hellman share should be μ bits and that the post-quantum KEMs should encapsulate uniform random bitstrings of length μ. Likewise, the pseudorandom functions should have output length μ.

Our security proofs rely on a number of standard assumptions such as the Decisional Diffie-Hellman (DDH) assumption, the existence of Pseudorandom Functions (PRF), and ciphertext indistinguishability under chosen plaintext attacks (IND-CPA) for key encapsulation mechanisms.

An adversary $\mathcal{A}$ has advantage

1. $\mathrm{Adv}^{\text{ddh}}_{\mathbf{G}}(\mathcal{A})$ of distinguishing (g^x, g^y, g^{xy}) from (g^x, g^y, g^w) for a random w;
2. $\mathrm{Adv}^{\text{prf}}_{\mathsf{Prf}}(\mathcal{A})$ of distinguishing Prf from a random function; and
3. $\mathrm{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{A})$ of violating IND-CPA security of Π.

We also rely on the less standard Oracle Diffie-Hellman (ODH) assumption for a group $\mathbf{G}$ and a hash function H. We will refer to a tuple $(g^u, g^v, h, \mathcal{H}_v)$ as an ODH instance. The ODH assumption is that distinguishing $\mathsf{H}(g^{uv})$ from a random string is hard given g^u, g^v, and an oracle, $\mathcal{H}_v$ that evaluates $\mathsf{H}(W^v)$ for arbitrary $W \neq g^u$. The ODH instance includes a value h that is generated as $\mathsf{H}(g^{uv})$ or as $h \xleftarrow{\$} \{0,1\}^\mu$ with equal probability. The oracle Diffie-Hellman assumption was introduced in [1], and variants have been used to prove the security of TLS ciphersuites in the ACCE model [18, 21]. An adversary's advantage in the ODH game is $\mathrm{Adv}^{\text{odh}}_{\mathbf{G},\mathsf{H}}(\mathcal{A})$. This advantage measure is well defined with respect to post-quantum adversaries, however whenever we evaluate it for a post-quantum adversary we will assume it is equal to 1.

We also require an extract-and-expand key derivation function and a stateful length-hiding authenticated encryption scheme.

2.6 Extract-and-expand key derivation functions

The extract-and-expand construction for key derivation functions was introduced by Krawczyk in [20]. Following the generic construction given in the same work,

we consider three functions: Kdf, a key derivation function; Prf, a pseudorandom function; and Xtr, a randomness extractor. For our application we will require that Xtr is a generic statistical randomness extractor. A formal definition can be found in [20], but this essentially means that it produces uniform random output from *every* source of sufficiently high min-entropy. Generic extractors require a salt input, so our Xtr will take the form:

$$\mathsf{Xtr} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^{\mu}.$$

The first input is a random non-secret salt, and the second input is secret key material. We keep this form in order to stay close to what would actually be deployed in practice, but we will ultimately model Xtr as a random oracle.

The function Kdf takes an extractor salt xts, context information ctx, secret key material skm, and the length of the desired key ℓ. The output K of $\mathsf{Kdf}(xts, ctx, skm, \ell)$ is computed as

1. $prk = \mathsf{Xtr}(xts, skm)$
2. $K = \mathsf{Prf}^*(prk, ctx, \ell)$.

Here Prf^* is an output feedback mode for Prf that enables variable-length expansion. For simplicity, the length parameters for Kdf and Prf^* will be omitted.

2.7 Stateful length-hiding authenticated encryption

A stateful encryption scheme StE is a tuple of algorithms $(\mathsf{StE.Init}, \mathsf{StE.Gen}, \mathsf{StE.Enc}, \mathsf{StE.Dec})$. The algorithm StE.Init takes no input and outputs the state (st_e, st_d) used by the encryption scheme. The algorithm StE.Gen samples a key K from the keyspace. The algorithm StE.Enc takes a secret key K, a ciphertext length ℓ, header data $h \in \{0,1\}^*$, a plaintext $m \in \{0,1\}^*$, and the current state st_e. It outputs a ciphertext $C \in \{0,1\}^{\ell}$ and the updated state st'_e. The encryption algorithm returns $\perp$ iff the message is of an invalid length. The algorithm StE.Dec takes a key K, header data h, a ciphertext C, and the current state st_d. It outputs the corresponding decryption m' of C and the updated state st_d.

A stateful encryption scheme is correct if any sequence of encryptions and decryptions

$$\left\{(C_i, st_e^i) = \mathsf{StE.Enc}(K, \ell_i, h_i, m_i, st_e^{i-1})\right\}_{1 \le i \le n},$$
$$\left\{(m'_i, st_d^i) = \mathsf{StE.Dec}(K, \ell_i, h_i, C_i, st_d^{i-1})\right\}_{1 \le i \le n},$$

Alg. 2.1 $\mathsf{Encrypt}(\pi_i^s, m_0, m_1, \mathtt{len}, H)$

$(C^{(0)}, st_e^{(0)}) = \mathsf{StE.Enc}(\pi_i^s.K_e, \mathtt{len}, H, m_0, \pi_i^s.st_e)$
$(C^{(1)}, st_e^{(1)}) = \mathsf{StE.Enc}(\pi_i^s.K_e, \mathtt{len}, H, m_1, \pi_i^s.st_e)$
if $C^{(0)} = \perp$ or $C^{(1)} = \perp$ **then**
  return $\perp$
end if
$\pi_i^s.u = \pi_i^s.u + 1$
$\pi_i^s.st_e = st_e^{(\pi_i^s.b)}$
$\pi_i^s.C[\pi_i^s.u] = C^{(\pi_i^s.b)}$
$\pi_i^s.H[\pi_i^s.u] = H$
return $\pi_i^s.C[\pi_i^s.u]$

for which no $C_i = \perp$, satisfies $m_i = m'_i$. Note that the sequences must start from $(st_e^0, st_d^0) = \mathsf{StE.Init}()$ and a valid key $K = \mathsf{StE.Gen}()$.

We will require the stateful encryption schemes used in hybrid to be secure with respect to the following stateful Length-Hiding Authenticated Encryption (sLHAE) experiment. The experiment is typically defined with respect to a stateful challenger, but we will give a description, specific to our environment, where the state is held by a pair of ACCE session oracles. More formal definitions may be found in [21].

When a session oracle π_i^s enters the `accept` state with a unique matching session π_j^t it sets:

- $(\pi_i^s.st_e^0, \pi_i^s.st_d^0) = \mathsf{StE.Init}()$,
- $\pi_i^s.u = 0$,
- $\pi_i^s.v = 0$,
- and initializes $\pi_i^s.C$ as an empty list.

Examining Algorithms 2.1 and 2.2 one can see that there are two distinct settings based on whether $\pi_i^s.b = 0$ or 1. These definitions of the Encrypt and Decrypt oracles make it such that:

- When $\pi_i^s.b = 0$ the adversary is given an encryption oracle for m_0 and a decryption oracle that always returns $\perp$.
- When $\pi_i^s.b = 1$ the adversary is given an encryption oracle for m_1 and a decryption oracle that returns the correct decryption only after `phase` is set to 1.

The sLHAE experiment allows an adversary $\mathcal{A}$ to make an arbitrary number of Encrypt queries (Algorithm 2.1) to π_i^s and an arbitrary number of Decrypt queries (Algorithm 2.2) to π_j^t before outputting a guess b' of $\pi_i^s.b$. The adversary's advantage is

$$\mathrm{Adv}^{\mathtt{sLHAE}}_{\mathsf{StE}}(\mathcal{A}) = \left|\Pr\left(b' = \pi_i^s.b\right) - 1/2\right|.$$

Alg. 2.2 $\mathsf{Decrypt}(\pi_i^s, C, H)$

Require: π_j^t is a unique matching session to π_i^s.
  if $\pi_i^s.b = 0$ **then**
    return $\perp$
  end if
  $\pi_i^s.v = \pi_i^s.v + 1$
  $(m, \pi_i^s.st_d) = \mathsf{StE.Dec}(\pi_i^s.K_d, H, C, \pi_i^s.st_d)$
  if $\pi_i^s.v > \pi_j^t.u$ or $C \neq \pi_j^t.C[\pi_i^s.v]$ or $H \neq \pi_j^t.H[\pi_i^s.v]$ **then**
    $\texttt{phase} = 1$
  end if
  if $\texttt{phase} = 1$ **then return** m **end if**
  return $\perp$

3 Protocols

3.1 Generic hybrid protocol

An instantiation of the hybrid protocol must specify:

- unique strings `proto_id`, `t_auth` = `proto_id:auth` and `t_key` = `proto_id:key`;
- an ordered set of 0 or more key encapsulation mechanisms

$$\vec{\mathbb{E}} = \{\mathsf{KEM}_1, \ldots, \mathsf{KEM}_{n_E}\}$$
$$\mathsf{KEM}_k = (\mathsf{KeyGen}_k, \mathsf{Encaps}_k, \mathsf{Decaps}_k)$$

  with message spaces $\{\mathbb{M}_k\}_{1 \leq k \leq n_E}$;
- a hash function H;
- a generic randomness extractor Xtr; and
- a pseudorandom function with variable length output Prf^*.

We will refer to a specific instantiation as $\mathsf{hybrid}(\vec{\mathbb{E}})$ when necessary. We assume that each client receives a certified copy of each server's long-term public key along with associated identity information. Let P_i be a client and P_j be a server with long-term DH key (a, A). We refer to P_j's identity information as its *identity digest* and denote it $\widehat{P_j}$. The following routines describe the actions taken by P_i and P_j in negotiating a key using $\mathsf{hybrid}(\vec{\mathbb{E}})$. Figure 3.1 provides a higher level description for the $n_E = 1$ case.

Client initialization

When π_i^s is directed to execute the hybrid protocol with P_j it:

1. Sets $\pi_i^s.\rho = \texttt{init}$, and $\pi_i^s.pid = j$
2. Checks the certificate for P_j's long-term key and aborts if it is invalid.
3. Generates an ephemeral DH keypair
$$(\pi_i^s.x, \pi_i^s.X) = \mathsf{DHGen}(1^\lambda).$$
4. Generates an ephemeral keypair for each KEM:
$$(\pi_i^s.esk_k, \pi_i^s.epk_k) = \mathsf{KeyGen}_k(1^\lambda)$$
  for $k \in \{1, \ldots, n_E\}$.
5. Outputs $(X, epk_1, \ldots, epk_{n_E})$.

Server response

On receipt of $(X, epk_1, \ldots, epk_{n_E})$, session π_j^t:

1. Sets $\pi_j^t.\rho = \texttt{resp}$, and $\pi_j^t.pid = \perp$.
2. Generates an ephemeral DH keypair
$$(\pi_j^t.y, \pi_j^t.Y) = \mathsf{DHGen}(1^\lambda).$$
3. Computes the Diffie-Hellman portions of the pre-master secret:
$$\pi_j^t.s_0 = \mathsf{H}(X^a); \quad \pi_j^t.s_1 = X^y.$$
4. Encapsulates a random value from the message space of each KEM:
$$\pi_j^t.s_{k+1} \xleftarrow{\$} \mathbb{M}_k \text{ for } k \in \{1, \ldots, n_E\}$$
$$\pi_j^t.ct_k = \mathsf{Encaps}_k(s_{k+1}, epk_k).$$
5. Forms the pre-master secret
$$\pi_j^t.pms = s_0 || s_1 || \ldots || s_{n_E+1}.$$
6. Forms the extraction salt
$$\pi_j^t.T = \widehat{P_j} || A || X || Y || epk_1 || \ldots || epk_{n_E} || ct_1 || \ldots || ct_{n_E}.$$
7. Extracts a pseudorandom key from pms:
$$\pi_j^t.prk = \mathsf{Xtr}(T, pms).$$
8. Computes the authentication tag:
$$\pi_j^t.auth = \mathsf{Prf}^*(prk, \texttt{t\_auth}).$$
9. Sets the session key
$$\pi_j^t.K = \mathsf{Prf}^*(prk, \texttt{t\_key}).$$
10. Sets
$$\pi_j^t.sid = (X, epk_1, \ldots, epk_{n_E}, Y, ct_1, \ldots, ct_{n_E}, auth)$$
11. Sets $\pi_j^t.\alpha = \texttt{accept}$.
12. Outputs $(Y, ct_1, \ldots, ct_{n_E}, auth)$.

Anonymous client		Server with long-term DH key (a, A) and identity digest $\widehat{P_j}$
$(x, X) = \mathsf{DHGen}(1^\lambda)$		
$(esk_1, epk_1) = \mathsf{KeyGen}_1(1^\lambda)$		
	$\xrightarrow{X, epk_1}$	
		$(y, Y) = \mathsf{DHGen}(1^\lambda)$
		$s_0 = \mathsf{H}(X^a)$
		$s_1 = X^y$
		$s_2 \xleftarrow{\$} \mathbb{M}_1$
		$ct_1 = \mathsf{Encaps}_1(s_2, epk_1)$
		$pms = s_0 \| s_1 \| s_2$
		$T = \widehat{P_j} \| A \| X \| Y \| epk_1 \| ct_1$
		$prk = \mathsf{Xtr}(T, pms)$
		$auth = \mathsf{Prf}^*(prk, \texttt{t\_auth})$
	$\xleftarrow{Y, ct_1, auth}$	
$s_0 = \mathsf{H}(A^x)$		
$s_1 = Y^x$		
$s_2 = \mathsf{Decaps}_1(ct_1, esk_1)$		
$pms = s_0 \| s_1 \| s_2$		
$T = \widehat{P_j} \| A \| X \| Y \| epk_1 \| ct_1$		
$prk = \mathsf{Xtr}(T, pms)$		
ensure $auth = \mathsf{Prf}^*(prk, \texttt{t\_auth})$		
$K = \mathsf{Prf}^*(prk, \texttt{t\_key})$		$K = \mathsf{Prf}^*(prk, \texttt{t\_key})$

Fig. 3.1. The proposed protocol with a single KEM.

13. Erases all temporary values and session state not required by the security experiment.

Remark 3.1. Steps 7 and 8 are equivalent to

$$\pi_j^t.auth = \mathsf{Kdf}(T, \texttt{t\_auth}, pms),$$

and Steps 7 and 9 are equivalent to

$$\pi_j^t.K = \mathsf{Kdf}(T, \texttt{t\_key}, pms).$$

Client finish

On receipt of $(Y, ct_1, \ldots, ct_{n_E}, auth)$, session π_i^s:

1. Computes the Diffie-Hellman portions of the premaster secret:
$$\pi_i^s.s_0 = \mathsf{H}(A^x); \quad \pi_i^s.s_1 = Y^x.$$
2. Decapsulates ct_1 through ct_{n_E}:
$$\pi_i^s.s_{k+1} = \mathsf{Decaps}_k(ct_k, esk_k) \text{ for } k \in \{1, \ldots, n_E\}$$
3. Forms $\pi_i^s.pms$ and $\pi_i^s.T$ as above.
4. Checks that $\mathsf{Kdf}(T, \texttt{t\_auth}, pms)$ matches the received authentication tag and aborts with $\pi_i^s.\alpha =$ `reject` if it does not.
5. Sets $\pi_i^s.K = \mathsf{Kdf}(T, \texttt{t\_key}, pms)$.
6. Sets
$$\pi_i^s.sid = (X, epk_1, \ldots, epk_{n_E}, Y, ct_1, \ldots, ct_{n_E}, auth).$$
7. Sets $\pi_i^s.\alpha =$ `accept`.
8. Erases all temporary values and session state not required by the security experiment.

3.2 The case of zero KEMs

The hybrid protocol with zero additional KEMs, $n_E = 0$, is almost identical to ntor: the client sends (X), and the server responds with $(Y, auth)$. Figure 3.2 provides a high level description of ntor for comparison with Figure 3.1. The most obvious difference is that the premaster secret in ntor contains g^{xa}, whereas the premaster secret in hybrid contains $\mathsf{H}(g^{xa})$. This is an artifact of our reduction from the Oracle Diffie-Hellman problem (see Section 4.1).

There are a few important differences regarding the use of the functions Xtr and Prf. These have direct analogues in the deployed instantiation of ntor, but are

Anonymous client		Server with long-term DH key (a, A) and identity digest $\widehat{P_j}$
$(x, X) = \mathsf{DHGen}(1^\lambda)$		
	$\xrightarrow{X}$	
		$(y, Y) = \mathsf{DHGen}(1^\lambda)$
		$pms = X^y \vert\vert X^a$
		$T_1 = \widehat{P_j} \vert\vert A \vert\vert X \vert\vert Y \vert\vert \texttt{proto\_id}$
		$T_2 = \widehat{P_j} \vert\vert A \vert\vert Y \vert\vert X \vert\vert \texttt{proto\_id} \vert\vert \texttt{Server}$
		$vk = \mathsf{HMAC}(\texttt{t\_verify}, pms \vert\vert T_1)$
		$auth = \mathsf{HMAC}(\texttt{t\_mac}, vk \vert\vert T_2)$
	$\xleftarrow{Y, auth}$	
$pms = Y^x \vert\vert A^x$		
$T_1 = \widehat{P_j} \vert\vert A \vert\vert X \vert\vert Y \vert\vert \texttt{proto\_id}$		
$T_2 = \widehat{P_j} \vert\vert A \vert\vert Y \vert\vert X \vert\vert \texttt{proto\_id} \vert\vert \texttt{Server}$		
$vk = \mathsf{HMAC}(\texttt{t\_verify}, pms \vert\vert T_1)$		
ensure $auth = \mathsf{HMAC}(\texttt{t\_mac}, vk \vert\vert T_2)$		
$prk = \mathsf{HMAC}(\texttt{t\_key}, pms \vert\vert T_1)$		$prk = \mathsf{HMAC}(\texttt{t\_key}, pms \vert\vert T_1)$
$K = \mathsf{HMAC}^*(prk, \texttt{m\_expand})$		$K = \mathsf{HMAC}^*(prk, \texttt{m\_expand})$

Fig. 3.2. The ntor protocol (as described in Tor proposal #216 [23]).

treated as random oracles in [13]. The formal description of ntor specifies the use of three distinct hash functions, but the proof does not specify any concrete requirements for the hash functions to satisfy. In the engineering specification [23] the hash functions are replaced by a single pseudorandom function with three distinct non-secret keys. In the deployed implementation, the pseudorandom function is instantiated with HMAC-SHA256 and the keys are taken to be ASCII strings that include a protocol identifier and the intended use for the hash value, e.g.

`ntor-curve25519-sha256-1:key_extract`.

The engineering specification also switches from a hash function to HKDF-SHA256 for the key derivation function. The end result of these modifications is that the authentication tag and key are computed as:

$$
\begin{aligned}
pms &= g^{xy} \vert\vert g^{xa} \\
T_1 &= \widehat{P_j} \vert\vert A \vert\vert X \vert\vert Y \vert\vert \texttt{proto\_id} \\
T_2 &= \widehat{P_j} \vert\vert A \vert\vert Y \vert\vert X \vert\vert \texttt{proto\_id} \vert\vert \texttt{Server} \\
vk &= \mathsf{HMAC\text{-}SHA256}(\texttt{proto\_id:verify}, pms \vert\vert T_1) \\
auth &= \mathsf{HMAC\text{-}SHA256}(\texttt{proto\_id:mac}, vk \vert\vert T_2) \\
prk &= \mathsf{HMAC\text{-}SHA256}(\texttt{proto\_id:key\_extract}, pms \vert\vert T_1) \\
K &= \mathsf{HMAC\text{-}SHA256}^*(prk, \texttt{proto\_id:key\_expand})
\end{aligned}
$$

The use of fixed HMAC keys across all sessions and the ad-hoc concatenation of public and private material (e.g. $pms \vert\vert T_1$) forgoes the security guarantees that come from using a PRF instead of a hash function. In essence, this usage treats HMAC as if it were a random oracle and ignores the security proofs and usage guidelines of this function.

Contrast this with the variant of ntor that we obtain through our hybrid protocol. If HKDF-SHA256 is used for Kdf, then we have Xtr = HMAC-SHA256 and Prf = HMAC-SHA256. The authentication tag and key are computed as:

$$
\begin{aligned}
pms &= \mathsf{H}(g^{xa}) \vert\vert g^{xy} \\
T &= \widehat{P_j} \vert\vert A \vert\vert X \vert\vert Y \\
prk &= \mathsf{HMAC\text{-}SHA256}(T, pms) \\
auth &= \mathsf{HMAC\text{-}SHA256}^*(prk, \texttt{proto\_id:auth}) \\
K &= \mathsf{HMAC\text{-}SHA256}^*(prk, \texttt{proto\_id:key}).
\end{aligned}
$$

The main benefit of this approach is that it follows the recommended usage of HKDF [20]. The approach used in ntor is somewhat ad-hoc, but the use of T_2 prevents an interesting class of multi-user attacks on vk. In Section 6 we discuss Zaverucha's multi-user attack on extract-and-expand KDFs [30] and why the T_2 countermeasure can be omitted without seriously affecting security.

4 Security

We will now prove the security of the hybrid protocol in the SACCE model. Theorem 4.1 considers the case of zero KEMs, and Theorem 4.2 considers the case of a single KEM. Generalizations to arbitrary numbers of KEMs are immediate.

As discussed in Section 3.2, the hybrid protocol with zero KEMs is not equivalent to ntor. However, the two protocols are very similar, and our proof should provide some renewed confidence in ntor.

The theorems follow easily from Lemmas 4.3 and 4.4 of Section 4.1. These lemmas bound an arbitrary adversary's advantage in the s-acce-sa and s-acce-ae experiments respectively.

Theorem 4.1. *Let* $\mathbf{G}$ *be a finite abelian group,* H *be a cryptographic hash function,* Xtr *be a random oracle,* Prf *be a pseudorandom function, and* StE *be a stateful length-hiding authenticated encryption scheme. The* hybrid *protocol with zero KEMs is pre-quantum SACCE secure under the ODH assumption for* $(\mathbf{G}, \mathsf{H})$*, and the DDH assumption for* $\mathbf{G}$*.*

Theorem 4.2. *Let* $\mathbf{G}$ *be a finite abelian group,* H *be a cryptographic hash function,* Xtr *be a random oracle,* Prf *be a pseudorandom function, and* StE *be a stateful length-hiding authenticated encryption scheme. The* hybrid *protocol with one post-quantum KEM provides transitional SACCE security under the ODH assumption for* $(\mathbf{G}, \mathsf{H})$*, and the assumption of IND-CPA security for* KEM_1*. Furthermore, the* hybrid *protocol with one pre-quantum KEM is pre-quantum SACCE secure under the ODH assumption for* $(\mathbf{G}, \mathsf{H})$ *and* either *the DDH assumption in* $\mathbf{G}$ or *the IND-CPA security of* KEM_1*.*

4.1 Authentication

We now bound an arbitrary adversary's advantage in the s-acce-ae experiment. Let n_P be an upper bound on the number of parties that the adversary initiates, and n_S be an upper bound on the number of sessions. Lemma 4.3 holds independently of the number and types of additional KEMs used, so we will simply refer to the scheme as hybrid.

Lemma 4.3. *Let* H*,* Xtr *and* Prf *be the functions specified by* hybrid*. If* Xtr *is a generic statistical randomness extractor, then for any adversary* $\mathcal{A}$ *there exist algorithms* $\mathcal{B}_0$ *and* $\mathcal{B}_1$*, each with running time that is an additive constant greater than that of* $\mathcal{A}$*, such that:*

$$\mathrm{Adv}^{\texttt{s-sacce-sa}}_{\mathsf{hybrid}}(\mathcal{A}) \leq n_P \,\mathrm{Adv}^{\mathrm{odh}}_{\mathbf{G},\mathsf{H}}(\mathcal{B}_0) + \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{Prf}}(\mathcal{B}_1) + 2^{-\mu}.$$

Proof. Let $\mathsf{break}^{(0)}_{\delta}$ be the event that a client session accepts maliciously in Game δ with $\mathcal{A}$ as the adversary. Recall that in the s-sacce-sa experiment the adversary commits to an index, (i^*, s^*), of a session that it will cause to accept maliciously. Let P_j be the party designated by $\pi^{s^*}_{i^*}.pid$. Note that the adversary may deliver the outgoing message of $\pi^{s^*}_{i^*}$ to more than one session controlled by P_j. Let $S \subseteq \{1, \ldots, n_S\}$ be the index set of P_j's sessions to which the adversary delivers the outgoing message of $\pi^{s^*}_{i^*}$.

Game 1.
This game is identical to the selective SACCE authentication experiment, hence

$$\mathrm{Adv}^{\texttt{s-sacce-sa}}_{\mathsf{hybrid}}(\mathcal{A}) = \Pr\left(\mathsf{break}^{(0)}_1\right). \tag{1}$$

Game 2.
The challenger proceeds as in Game 1, but replaces the value $s_0 = \mathsf{H}(A^x)$ in session $\pi^{s^*}_{i^*}$ with $\widetilde{s_0} \xleftarrow{\$} \{0,1\}^\mu$. The challenger also replaces $\mathsf{H}(X^a)$ with $\widetilde{s_0}$ in session π^t_j for $t \in S$.

Algorithm $\mathcal{B}_0$ takes an ODH instance $(U = g^u, V = g^v, h, \mathcal{H}_v)$ as input and simulates a challenger in Game 1 against adversary $\mathcal{A}$. It behaves as follows:

- At the beginning of the game, it sets the long-term public key of a randomly chosen party to V. Call this party P_{j^*}.
- It aborts if $\pi^{s^*}_{i^*}.pid \neq j^*$.
- It aborts if the adversary corrupts P_{j^*}.
- It sets $\pi^{s^*}_{i^*}.X = U$ and $\pi^{s^*}_{i^*}.s_0 = h$.
- It uses $\mathcal{H}_v$ and h to simulate knowledge of P_{j^*}'s long-term secret in any session, π^t_j, for which $\pi^t_j.\rho = \texttt{resp}$. If $\pi^t_j.X = U$ the challenger sets $\pi^t_j.s_0 = h$; if $\pi^t_j.X \neq U$ the challenger sets $\pi^t_j.s_0 = \mathcal{H}_v(X)$.

Suppose that $\mathcal{B}_0$ does not abort. If $h = \mathsf{H}(g^{uv})$, the algorithm provides a faithful simulation of Game 1. Otherwise, h is uniformly random and the adversary's view is of Game 2. Hence, an adversary's advantage in distinguishing between these two games (when $\mathcal{B}_0$ does not abort) can be no greater than $\mathcal{B}_0$'s advantage in the ODH game.

The possibility that $\mathcal{B}_0$ aborts due to a $\mathsf{Corrupt}(P_j)$ query does not change the probability that $\pi_{i*}^{s^*}$ accepts maliciously in Game 2. Such a corruption either occurs after $\pi_{i*}^{s^*}$ accepts, or precludes the possibility of $\pi_{i*}^{s^*}$ accepting maliciously. The possibility that $\mathcal{B}_0$ aborts due to $\pi_{i*}^{s^*}.pid \neq j$, however, reduces its advantage in the ODH experiment by a factor of n_P. Hence,

$$\Pr\left(\mathsf{break}_1^{(0)}\right) \leq \Pr\left(\mathsf{break}_2^{(0)}\right) + n_P \operatorname{Adv}^{\mathrm{odh}}_{\mathbf{G},\mathsf{H}}(\mathcal{B}_0). \quad (2)$$

Game 3.
The challenger proceeds as in Game 2 for all sessions controlled by parties other than P_j, as well as for P_j's sessions not indexed by an element of S. For each $t \in S$ the challenger replaces $\pi_j^t.prk$ with $\widetilde{prk}_t \xleftarrow{\$} \{0,1\}^\mu$. Likewise if the adversary delivers the outgoing message of π_j^t to $\pi_{i*}^{s^*}$ the challenger replaces $\pi_{i*}^{s^*}.prk$ with $\widetilde{prk}_t$.

In Game 2 we ensure that the min-entropy of the input to Xtr is at least μ bits by choosing $\widetilde{s_0}$ randomly. Assuming that Xtr is a generic statistical randomness extractor each $\pi_j^t.prk$ is indistinguishable from a uniform random string in Game 2. In turn Game 3 is indistinguishable from Game 2, i.e.

$$\Pr\left(\mathsf{break}_2^{(0)}\right) = \Pr\left(\mathsf{break}_3^{(0)}\right). \quad (3)$$

Game 4.
The challenger proceeds as in Game 3 but aborts if the adversary delivers any of the outbound messages from $\{\pi_j^t : t \in S\}$ to $\pi_{i*}^{s^*}$. As each of these messages would cause $\pi_{i*}^{s^*}$ to accept non-maliciously, this change does not affect the probability that $\pi_{i*}^{s^*}$ accepts maliciously.

$$\Pr\left(\mathsf{break}_3^{(0)}\right) = \Pr\left(\mathsf{break}_4^{(0)}\right). \quad (4)$$

Game 5.
The challenger proceeds as in Game 4 but replaces Prf in session $\pi_{i*}^{s^*}$ with a random function F. In Game 4 the value $\pi_{i*}^{s^*}.prk$ is uniformly random. Consequently, if $\mathcal{A}$ can distinguish Game 5 from Game 4, then there exists an algorithm $\mathcal{B}_1$ that runs in essentially the same time that can answer a PRF challenge.

Specifically, we construct an algorithm $\mathcal{B}_1$ that solves a PRF challenge for the string `t_auth`. It takes as input $z \in \{0,1\}^\mu$ that is promised to have been generated either as $z \xleftarrow{\$} \{0,1\}^\mu$ or as $z = \mathsf{Prf}(k, \mathtt{t\_auth})$ for some $k \xleftarrow{\$} \{0,1\}^\mu$. It then simulates a challenger in Game 4 against adversary $\mathcal{A}$, and sets $\pi_{i*}^{s^*}.auth = z$.

If z was generated as $\mathsf{Prf}(k, \mathtt{t\_auth})$ then the adversary's view is identical to Game 4. Otherwise z was generated randomly and the adversary's view is of Game 5. Thus $\mathcal{B}_1$ wins the PRF game with advantage equal to the game distinguisher's advantage:

$$\Pr\left(\mathsf{break}_4^{(0)}\right) \leq \Pr\left(\mathsf{break}_5^{(0)}\right) + \operatorname{Adv}^{\mathrm{prf}}_{\mathsf{Prf}}(\mathcal{B}_1). \quad (5)$$

Final analysis.
The only way the adversary can cause $\pi_{i*}^{s^*}$ to accept without a matching session in Game 5 is to guess the value of $F(\widetilde{prk}, \mathtt{t\_auth})$. Since F is a random function they are successful with probability $2^{-\mu}$, hence

$$\Pr\left(\mathsf{break}_5^{(0)}\right) = 2^{-\mu}. \quad (6)$$

This establishes the claim. □

4.2 Channel security

We now bound an arbitrary adversary's advantage in the s-sacce-ae experiment. Again let n_P be an upper bound on the number of parties that the adversary initiates, and n_S be an upper bound on the number of sessions. Let n_E be the number of KEMs used in the hybrid protocol and let $\vec{\mathbb{E}} = (\mathsf{KEM}_1, \ldots, \mathsf{KEM}_{n_E})$.

Lemma 4.4. *Let* Xtr, Prf, *and* StE *be the functions specified by* $\mathsf{hybrid}(\vec{\mathbb{E}})$. *If* Xtr *is a generic statistical randomness extractor, then for any adversary,* $\mathcal{A}$, *there exist algorithms* $\mathcal{B}_{2.0}, \mathcal{B}_{2.1} \ldots, \mathcal{B}_{2.n_E}, \mathcal{B}_4$, *and* $\mathcal{B}_5$, *each with running time that is an additive constant greater than that of* $\mathcal{A}$, *such that*

$$\operatorname{Adv}^{\text{s-sacce-ae}}_{\mathsf{hybrid}(\vec{\mathbb{E}})}(\mathcal{A}) \leq \operatorname{Adv}^{\mathrm{prf}}_{\mathsf{Prf}}(\mathcal{B}_4) + \operatorname{Adv}^{\mathrm{sLHAE}}_{\mathsf{StE}}(\mathcal{B}_5) + \min\left\{\operatorname{Adv}^{\mathrm{ddh}}_{\mathbf{G}}(\mathcal{B}_{2.0}), \min_{1\leq k\leq n_E}\left\{\operatorname{Adv}^{\text{ind-cpa}}_{\mathsf{KEM_k}}(\mathcal{B}_{2.k})\right\}\right\}$$

Proof. Recall that in the s-acce-ae experiment the adversary commits to a matching pair of sessions, $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$, in which the initiator does not accepted maliciously. We assume, without loss of generality, that $\pi_{i*}^{s^*}$ has the role of initiator. The adversary's output is either (j^*, t^*, b^*) or (i^*, s^*, b^*).

Let $\mathsf{break}_\delta^{(1)}$ be the event that $\mathcal{A}$ answers the encryption challenge successfully in Game δ, e.g. that $\mathcal{A}$ output (i^*, s^*, b^*) and $b^* = \pi_{i*}^{s^*}.b$.

Game 1.
This game is identical to the selective SACCE channel security experiment. By definition,

$$\mathrm{Adv}^{\text{s-sacce-ae}}_{\mathsf{hybrid}(\vec{\mathbb{E}})}(\mathcal{A}) = \left|\Pr\left(\mathsf{break}_1^{(1)}\right) - 1/2\right|. \quad (7)$$

We now consider n_E+1 games that each replace one share of the premaster secret (one of $s_1, s_2, \ldots, s_{n_E+1}$) with a uniform random value from the appropriate sample space. We show that an advantage in distinguishing any of these games from Game 1 implies a corresponding advantage in either an IND-CPA or a DDH challenge. To simplify our proof, we assume that s_1 is the only secret protected by the DDH assumption and we treat it separately in Game 2.0.

Game 2.0
The challenger proceeds as in Game 1, however when it simulates sessions $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ it replaces s_1 with $\tilde{s}_1 \xleftarrow{\$} \mathbf{G}$.

The algorithm $\mathcal{B}_{2.0}$ interpolates between Games 1 and 2.0 by embedding the values from a DDH challenge $(\tilde{X}, \tilde{Y}, \tilde{U})$ as $\pi_i^s.X, \pi_j^t.Y$ and s_1 respectively. This gives us

$$\Pr\left(\mathsf{break}_1^{(1)}\right) \leq \Pr\left(\mathsf{break}_{2.0}^{(1)}\right) + \mathrm{Adv}^{\text{ddh}}_{\mathbf{G}}(\mathcal{B}_{2.0}). \quad (8)$$

Game 2.k for $k \in \{1, \ldots, n_E\}$
When the challenger simulates session $\pi_{j^*}^{t^*}$ it replaces ciphertext ct_k with one that is unrelated to s_{k+1}. Specifically it generates s_{k+1} as usual but samples a second random value $\widetilde{s_{k+1}} \xleftarrow{\$} \mathbb{M}_k$. It then sets $\pi_{j^*}^{t^*}.ct_k = \widetilde{ct}$ where $\widetilde{ct} = \mathsf{Encaps}_k(\widetilde{s_{k+1}}, epk_k)$. In session $\pi_{i^*}^{s^*}$ it sets $\pi_{i^*}^{s^*}.s_{k+1} = s_{k+1}$ and discards the value obtained through decapsulation.

The algorithm $\mathcal{B}_{2.k}$ interacts with an IND-CPA challenger for KEM_k. It receives a public key $\widetilde{epk}$ from the challenger, samples m_0 and m_1 uniformly from $\mathbb{M}_k$, and then requests an IND-CPA challenge for m_0 and m_1. It receives $\widetilde{ct}$ that is promised to be an encapsulation of either m_0 or m_1. It then simulates the challenger in Game 1 with adversary $\mathcal{A}$ as follows:

- It samples a uniform bit u.
- In session $\pi_{i^*}^{s^*}$ it replaces $\pi_{i^*}^{s^*}.epk_k$ with $\widetilde{epk}$, and replaces $\pi_{i^*}^{s^*}.esk_i$ with $\perp$.
- In session $\pi_{j^*}^{t^*}$ it replaces ct_k with $\widetilde{ct}$.
- In both $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ it sets $s_{k+1} = m_u$.

If $\widetilde{ct}$ is an encapsulation of m_u then $\mathcal{B}_{2.k}$ provides a faithful simulation of Game 1. Otherwise $\widetilde{ct}$ is an encapsulation of m_{1-u} and the adversary's view is of Game 2.k.

Since Xtr is a generic statistical extractor, prk is identically distributed in Games 1 and 2.k. This prevents the adversary from distinguishing the two games via $auth$ or K. An adversary with non-negligible advantage in distinguishing Games 1 and 2.k must detect the difference in ct_k. Hence the adversary's advantage translates to an equivalent advantage for $\mathcal{B}_{2.k}$ in the IND-CPA challenge. This implies

$$\Pr\left(\mathsf{break}_1^{(1)}\right) \leq \Pr\left(\mathsf{break}_{2.k}^{(1)}\right) + \mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{KEM_k}}(\mathcal{B}_{2.k}) \quad \forall k. \quad (9)$$

Game 3.
The challenger executes some variant of Game 2 but replaces the pseudorandom key, $prk = \mathsf{Xtr}(T, pms)$, in sessions $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ with a uniform random value $\widetilde{prk} \in \{0,1\}^\mu$.

In each variant of Game 2 we ensure that the input to Xtr has min-entropy of at least μ bits. Assuming that Xtr is a generic statistical randomness extractor, the value of prk is uniformly random in each variant of Game 2. Hence this game is indistinguishable from the chosen variant of Game 2, and we obtain

$$\Pr\left(\mathsf{break}_{2.k}^{(1)}\right) = \Pr\left(\mathsf{break}_3^{(1)}\right) \quad \forall k. \quad (10)$$

Consequently the n_E inequalities of Equation 9 differ only in the magnitude of the adversary's advantage in the respective KEM game. This gives us

$$\Pr\left(\mathsf{break}_1^{(1)}\right) \leq \Pr\left(\mathsf{break}_3^{(1)}\right) + \min\left\{\mathrm{Adv}^{\text{ddh}}_{\mathbf{G}}(\mathcal{B}_{2.0}), \min_k\left\{\mathrm{Adv}^{\text{ind-cpa}}_{\mathsf{KEM_k}}(\mathcal{B}_{2.k})\right\}\right\}. \quad (11)$$

Game 4.
The challenger proceeds as in Game 3. In sessions $\pi_{i^*}^{s^*}$ and $\pi_{j^*}^{t^*}$ the challenger replaces Prf with a random function F. If $\mathcal{A}$ can distinguish Game 4 from Game 3, then there exists an algorithm $\mathcal{B}_4$, that runs in essentially the same time, that breaks the pseudorandomness of Prf. Hence,

$$\Pr\left(\mathsf{break}_3^{(1)}\right) \leq \Pr\left(\mathsf{break}_4^{(1)}\right) + \mathrm{Adv}^{\text{prf}}_{\mathsf{Prf}}(\mathcal{B}_4). \quad (12)$$

Game 5.
In this final game the challenger plays parallel roles as the s-acce-ae challenger from Game 4 and as two sLHAE challengers. It embeds the sLHAE games in the s-acce-ae simply by setting the keys used by $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$ appropriately.

The adversary's access to $\pi_{i*}^{s^*}.b$ and $\pi_{j*}^{t^*}.b$ is only through the Encrypt and Decrypt queries that define the sLHAE game. Hence the adversary outputs a $b^* = \pi_{i*}^{s^*}.b$ with advantage that is, by definition, no better than its advantage in the sLHAE game.

In Game 4 the session key K is uniformly random and independent of the messages exchanged between $\pi_{i*}^{s^*}$ and $\pi_{j*}^{t^*}$. As such, the adversary cannot detect the game transition. Letting $\mathcal{B}_5$ denote the challenger in Game 5, we have

$$1/2 \leq \Pr\left(\mathsf{break}_4^{(1)}\right) \leq 1/2 + \mathrm{Adv}_{\mathsf{StE}}^{\mathsf{sLHAE}}(\mathcal{B}_5). \qquad (13)$$

Combining the above inequalities establishes the claim □

4.3 Proofs of main theorems

Of Theorem 4.1. Modeling Xtr as a random oracle ensures that it is a generic statistical extractor [11, 20], so we may apply the bounds on $\mathrm{Adv}_{\mathsf{hybrid}}^{\text{s-sacce-sa}}(\cdot)$ and $\mathrm{Adv}_{\mathsf{hybrid}(\emptyset)}^{\text{s-sacce-ae}}(\cdot)$ from Lemmas 4.3 and 4.4.

Pre-quantum SACCE security only considers PPT pre-quantum adversaries $\mathcal{A}$. Hence the ODH assumption implies that $\mathrm{Adv}_{\mathsf{hybrid}}^{\text{s-sacce-sa}}(\mathcal{A})$ is negligible and the DDH assumption implies that $\mathrm{Adv}_{\mathsf{hybrid}(\emptyset)}^{\text{s-sacce-ae}}(\mathcal{A})$ is negligible. An application of Lemma 2.11 establishes the claim. □

Of Theorem 4.2. Modeling Xtr as a random oracle ensures that it is a generic statistical extractor, so we may apply the bounds on $\mathrm{Adv}_{\mathsf{hybrid}(\vec{\mathbb{E}})}^{\text{s-sacce-sa}}(\cdot)$ and $\mathrm{Adv}_{\mathsf{hybrid}(\vec{\mathbb{E}})}^{\text{s-sacce-ae}}(\cdot)$ from Lemmas 4.3 and 4.4.

Since transitional SACCE security forbids post-quantum adversaries in the pre-accept phase, the ODH assumption for $(\mathbf{G}, \mathsf{H})$ and Lemma 4.3 imply that $\mathrm{Adv}_{\mathsf{hybrid}}^{\text{s-sacce-sa}}(\mathcal{A})$ is bounded from above by a negligible quantity.

On the other hand, we must allow post-quantum adversaries, $\mathcal{Q}$, when applying Lemma 4.4. The DDH assumption does not hold against post-quantum adversaries, hence:

$$\min\left\{\mathrm{Adv}_{\mathbf{G}}^{\text{ddh}}(\mathcal{B}_{2.0}^{\mathcal{Q}}), \mathrm{Adv}_{\mathsf{KEM}_1}^{\text{ind-cpa}}(\mathcal{B}_{2.1}^{\mathcal{Q}})\right\} = \mathrm{Adv}_{\mathsf{KEM}_1}^{\text{ind-cpa}}(\mathcal{B}_{2.1}^{\mathcal{Q}}),$$

and the claim about transitional security follows.

The claim about pre-quantum security follows by re-evaluating the above minimization over PPT pre-quantum adversaries. □

5 Implementation and performance characteristics

We have implemented our protocol with HKDF-SHA256, curve25519 and ntruees443ep1. We have integrated this implementation into Tor 0.2.6.2-alpha and made it publicly available [28].

It was argued in [14] that ntruees443ep1 is IND-CCA2 secure against post-quantum adversaries at the $\lambda = 128$ level. This instantiation of hybrid is therefore estimated to provide transitional security at the $\lambda = 128$ bit security level.

Preliminary benchmarks comparing our instantiation's performance with that of ntor and that of the legacy Tor handshake (tap) are presented in Table 1.1. The benchmark was conducted on an Intel Core i7-2640M CPU clocked at 2.80GHz with TurboBoost disabled. RSA and $\mathbb{Z}_p^*$ Diffie-Hellman operations for tap were provided by OpenSSL 1.0.1i. The ECDH operations for ntor and hybrid were performed by the donna_c64 implementation of curve25519 from NaCL-20110221 [4].

The ntruees443ep1 operations were provided by libntruencrypt version 1.0.1 [17]. We note that libntruencrypt is a reference implementation. It does not attempt to be constant time and does not perform parameter set specific optimizations. Our libntruencrypt was compiled without vectorized convolutions. A previous version of this paper reported numbers with vectorized convolutions, and other optimizations. We have reported the unoptimized figures here as many Tor routers do not support SSSE3 operations.

Note also that ntruees443ep1 is designed to meet the requirements of a IND-CCA2 public key encryption scheme, but our setting only requires an IND-CPA KEM. An IND-CPA variant of NTRUEncrypt may be more efficient than ntruees443ep1.

The data in Tables 1.1 and 5.1 was gathered using Tor's internal benchmarking utility. Table 1.1 reports times averaged over 4096 trials for tap, ntor, hybrid, and the protocol of Ghosh-Kate. Communication costs are also reported in table 1.1. Table 5.1 reports first, second, and third quartiles for the hybrid protocol in a separate run of 4096 trials.

	hybrid		
	quartile	median	quartile
Client init	657 μs	661 μs	666 μs
Server response	304 μs	304 μs	305 μs
Client finish	217 μs	217 μs	217 μs

Table 5.1. Timing data for the hybrid protocol phases over 4096 samples.

6 Other security considerations

Multi-session attacks
The quadratic dependence on the number of sessions in the reduction from sacce-ae to s-sacce-ae in Lemma 2.11 suggests there might be a powerful multi-session attack that our security proof misses. Indeed, Zaverucha's attack [30] on extract-and-expand key derivation functions applies to our use of Kdf.

Consider an adversary that attempts to find a collision in an enormous collection of *auth* tags. In hybrid, a collision in *auth* tags suggests a collision in *prk* and hence in K. The adversary can Reveal one session of a colliding pair and hope to learn the session key of the second (this fails if distinct *prk* led to a collision in *auth*). This attack does not show up in our security proof as the selective security experiment is effectively single-session.

In ntor the *auth* tag is derived in a session-dependent manner that prevents this collision attack. We choose a different countermeasure and simply require *auth* to be of length $\mu = 2\lambda$. This is sufficient since the above collision attack on a random function of output length 2λ has cost 2^{λ}. Taking *auth* to be length 2λ is already required in the (single-session) transitional and post-quantum settings since quantum search provides a single-session preimage attack that would reveal *prk* for cost $\Theta(2^{\mu/2})$.

Forward secrecy
Any protocol meeting our definition of channel security provides forward secrecy. Our transitional SACCE protocols provide forward secrecy and resist adversaries that obtain quantum capabilities after session negotiation.

One-way anonymity
The ntor paper formalizes a notion of one-way anonymity that ntor satisfies. Their definition makes use of features of the AKE execution environment that are not paralleled by our ACCE execution environment, so we cannot apply their proof directly.

It is difficult to imagine a notion of one-way anonymity that would distinguish between our hybrid protocol and ntor operated in isolation. However, it is apparent that allowing multiple handshake types could have an impact on anonymity. In particular, if clients run different versions of the Tor software, then the set of ciphersuites they support may provide a mechanism for deanonymizing them.

The one-way anonymity notion proposed in [13] ignores the negotiation protocol and, therefore, cannot capture this subtlety. We leave it to future work to propose a stronger definition of one-way anonymity.

Multi-ciphersuite security
Bergsma, Dowling, Kohlar, Schwenk, and Stebila have shown that SSH is a secure ACCE protocol in a *multi-ciphersuite* setting that allows for long-term key reuse between ciphersuites [2]. A similar analysis will be necessary for Tor if servers are allowed to reuse their long-term keys between hybrid variants and ntor.

Post-quantum security
One could achieve post-quantum ACCE security by carefully incorporating a post-quantum signature (e.g. XMSS [8] or SPHINCS [3]), or a server-side static encryption key, into the handshake. Post-quantum authentication would also be necessary for key distribution. While ultimately necessary, integrating a new authentication mechanism would require significant modifications to the Tor protocol. We feel that doing so would decrease the likelihood of our protocol being deployed in a timely fashion.

Fully quantum ACCE environment
Our ACCE execution environment explicitly forbids quantum communication between the adversary and session oracles. We believe this is a natural restriction; we are only interested in proving that the hybrid protocol meets a reasonable definition of security when it is executed on a classical computer.

The assumption that the adversary does not perform *interactive* quantum computations with the session oracles is implicit in all related work.

Initial hash of static-ephemeral DH share
Our proof requires that the parties compute $s_0 = \mathsf{H}(g^{xa})$, rather than just g^{xa}, so that we can make use of the Oracle Diffie-Hellman assumption. The concatenation of the pre-master secret shares prevents us from

applying an ODH-like assumption directly to the extractor.

The ODH assumption is needed to ensure that the party in the responder role can respond consistently to multiple queries with the same X in Game 2 of Lemma 4.3. It is quite possible that computing s_0 as g^{xa} would have no practical security impact, yet we cannot prove this without additional assumptions.

Use of a statistical extractor
We have simplified our security proof by requiring statistical extractors in Lemmas 4.3 and 4.4. A concrete instantiation of Xtr is unlikely to meet this stringent requirement. However, the requirement is easily met when Xtr is modeled as a random oracle. A random oracle is a statistical extractor provided that the inputs to it are conditionally independent of its outputs. This is trivially true even in the quantum-accessible random oracle model of [5], hence no additional work is needed to treat quantum-accessible random oracles.

A more satisfying proof would replace Lemmas 4.3 and 4.4 with variants that rely only on computational extractors such as those found in [20]. Many constructions of computational extractors still require a hash function modeled as a random oracle. The construction of a computational extractor in the quantum-accessible random oracle model has, to the best of our knowledge, not been given in the literature.

Stateful length-hiding authenticated encryption
Keys negotiated by the circuit extension handshake protocol are used in Tor's relay protocol. The relay protocol exchanges data in 509 byte segments called *relay cells*. Each relay cell contains metadata (5 bytes), a truncated MAC (4 bytes), the length of the payload (2 bytes), the payload (ℓ bytes), and null padding ($509 - 11 - \ell$ bytes).

Tor currently uses AES-128 in counter mode to encrypt relay cells. The MAC is computed over end-to-end communications as a running SHA1 digest. Only the clent and the used exit-node check the MAC; relay cells that are forwarded are not authenticated.

With only a 32 bit MAC, this scheme cannot provide cryptographic authenticity guarantees. There are unique challenges in migrating to proper authenticated encryption. For example, encrypt-then-mac schemes cannot be used as the message expansion from concatenated MACs would leak the circuit length.

There are several proposals currently under consideration within the Tor community that would improve the situation [22, 25]. Future work should consider whether any of these proposals meet our requirement of stateful length-hiding authenticated encryption, or whether there is another notion that is more appropriate for Tor.

It will also be necessary to migrate to a cipher with 256 bit keys for 128 bit transitional or post-quantum security.

7 Conclusion

We have presented a transitionally secure SACCE protocol and provided an implementation in a form that could be integrated into Tor easily. To assist with this integration, and to guide experimentation with other instantiations, we have published an engineering specification of hybrid as Tor proposal #263 [27].

There are several barriers to deployment that remain.

First, Tor circuit extension handshakes are limited to 505 bytes due to the current specification of the CREATE cell. Our instantiation using ntruees443ep1 requires 693 bytes for the client to server message and 673 bytes for the server to client message. A discussion with Nick Mathewson about the size limit on CREATE cells has resulted in Tor proposal #249 [24], which would remedy this problem.

Second, the ntruees443ep1 parameter set is covered by U.S. Patent Nos. 6081597 and 7031468 [15, 16]. While the patents are free to use within open source software, and libntruencrypt is distributed under the GPL, some users and software distributors may wish to avoid patented cryptography entirely[1].

Third, it remains to be seen whether our instantiation provides acceptable performance. The figures of Table 1.1 indicate that our instantiation is as computationally expensive as the original tap handshake and is much less compact. It is, however, more compact than the proposal from Ghosh and Kate.

Directions for future work include: performance analyses for hybrid when instantiated with different post-quantum KEMs (or with a faster and/or constant-time implementation of NTRUEncrypt); a study of one-way anonymity that addresses the points raised in Section 6; a proof of security for hybrid that does not depend on statistical extractors; and the development of post-quantum ACCE protocols.

1 U.S. Patent No. 6081597 covers the core NTRU functionality and expires on August 19, 2017. U.S. Patent No. 7031468 covers the product-form efficiency enhancement and expires on August 24, 2021.

Acknowledgements

We are very grateful to Nick Mathewson and other members of the Tor community for their input on Tor proposal #263. We also wish to thank: Aniket Kate for discussing potential improvements to our scheme, Douglas Stebila for several enlightening conversations, and the anonymous reviewers for their close readings and detailed recommendations.

References

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *Topics in Cryptology — CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001 San Francisco, CA, USA, April 8–12, 2001 Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[2] Florian Bergsma, Benjamin Dowling, Florian Kohlar, Jörg Schwenk, and Douglas Stebila. Multi-ciphersuite security of the secure shell (SSH) protocol. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 369–381, New York, NY, USA, 2014. ACM.

[3] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 2015.

[4] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. NaCL: Networking and cryptography library. http://nacl.cr.yp.to/, 2011.

[5] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, 2011.

[6] Dan Boneh and Richard J. Lipton. Quantum cryptanalysis of hidden linear functions. In Don Coppersmith, editor, *Advances in Cryptology 1981 – 1997: Electronic Proceedings and Index of the CRYPTO and EUROCRYPT Conferences 1981 – 1997*, volume 1440 of *Lecture Notes in Computer Science*, chapter CRYPTO '95, pages 424–437. Springer, 2001.

[7] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570, 2015.

[8] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS – A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 – December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2011.

[9] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. NIST Internal Report 8105. http://dx.doi.org/10.6028/NIST.IR.8105, February 2016.

[10] NSA Information Assurance Directorate. Commercial national security algorithm suite. https://www.iad.gov/iad/programs/iad-initiatives/cnsa-suite.cfm, August 2015.

[11] Yevgeniy Dodis, Rosario Gennaro, Johan Håstad, Hugo Krawczyk, and Tal Rabin. Randomness extraction and key derivation using the CBC, cascade and HMAC modes. In Matt Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2004.

[12] Satrajit Ghosh and Aniket Kate. Post-quantum forward-secure onion routing. In Tal Malkin, Vladimir Kolesnikov, Bishop Allison Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security: 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 263–286. Springer, 2015.

[13] Ian Goldberg, Douglas Stebila, and Berkant Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013.

[14] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. Cryptology ePrint Archive, Report 2015/708, 2015. http://eprint.iacr.org/2015/708.

[15] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. United States Patent: 6081597 - Public key cryptosystem method and apparatus. https://www.google.com/patents/US6081597, June 2000.

[16] Jeffrey Hoffstein and Joseph H. Silverman. United States Patent: 7031468 - Speed enhanced cryptographic method and apparatus. https://www.google.com/patents/US7031468, April 2006.

[17] Security Innovation. libntruencrypt: NTRUEncrypt reference implementation. https://github.com/NTRUOpenSourceProject/ntru-crypto, 2015. Version 1.0.1.

[18] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 273–293. Springer, 2012.

[19] Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DH and TLS-RSA in the standard model. Cryptology ePrint Archive, Report 2013/367, 2013. http://eprint.iacr.org/2013/367.

[20] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010: 30th Annual Cryptology*

Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648. Springer, 2010.

[21] Hugo Krawczyk, Kenneth G. Paterson, and Hoeteck Wee. On the security of the TLS protocol: A systematic analysis. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 429–448. Springer, 2013.

[22] Nick Mathewson. Tor proposal # 202: Two improved relay encryption protocols for Tor cells. In [26], path: root/proposals/202-improved-relay-crypto.txt, blob: 695df306.

[23] Nick Mathewson. Tor proposal #216: Improved circuit-creation key exchange. In [26], path: root/proposals/216-ntor-handshake.txt, blob: f76e81cd.

[24] Nick Mathewson. Tor proposal #249: Allow create cells with >505 bytes of handshake data. In [26], path: root/proposals/249-large-create-cells.txt, blob: e04b4c0c.

[25] Nick Mathewson. Tor proposal #261: AEZ for relay cryptography. In [26], path: root/proposals/261-aez-crypto.txt, blob: 14435e7c.

[26] The Tor Project. Torspec Git repository. https://gitweb.torproject.org/torspec.git.

[27] John M. Schanck, William Whyte, and Zhenfei Zhang. Tor proposal #263: Request to change key exchange protocol for handshake. In [26], path: root/proposals/263-ntru-for-pq-handshake.txt, blob: a6732b60.

[28] John M. Schanck, William Whyte, and Zhenfei Zhang. Implementation of the current proposal using NTRUEncrypt. https://github.com/NTRUOpenSourceProject/ntru-tor, July 2015.

[29] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134. IEEE Computer Society Press, 1994.

[30] G.M. Zaverucha. Hybrid encryption in the multi-user setting. Cryptology ePrint Archive, Report 2012/159, 2012. http://eprint.iacr.org/2012/159.