

Proof-of-Possession for KEM Certificates using Verifiable Generation

Tim Güneysu

Ruhr University Bochum, Horst Görtz
Institute for IT-Security & DFKI
GmbH, Cyber-Physical Systems
Bochum, Germany
tim.gueneyasu@rub.de

Mike Ounsworth

Entrust
Ottawa, Ontario, Canada
mike.ounsworth@entrust.com

Philip Hodges

University of Waterloo
Waterloo, Ontario, Canada
pwhodges@uwaterloo.ca

Douglas Stebila

University of Waterloo
Waterloo, Ontario, Canada
dstebila@uwaterloo.ca

Georg Land

Ruhr University Bochum, Horst Görtz
Institute for IT-Security & DFKI
GmbH, Cyber-Physical Systems
Bochum, Germany
georg.land@rub.de

Greg Zaverucha

Microsoft Research
Redmond, Washington, USA
gregz@microsoft.com

ABSTRACT

Certificate authorities in public key infrastructures typically require entities to prove possession of the secret key corresponding to the public key they want certified. While this is straightforward for digital signature schemes, the most efficient solution for public key encryption and key encapsulation mechanisms (KEMs) requires an interactive challenge-response protocol, requiring a departure from current issuance processes. In this work we investigate how to non-interactively prove possession of a KEM secret key, specifically for lattice-based KEMs, motivated by the recently proposed KEMTLS protocol which replaces signature-based authentication in TLS 1.3 with KEM-based authentication. Although there are various zero-knowledge (ZK) techniques that can be used to prove possession of a lattice key, they yield large proofs or are inefficient to generate. We propose a technique called *verifiable generation*, in which a proof of possession is generated at the same time as the key itself is generated. Our technique is inspired by the Picnic signature scheme and uses the multi-party-computation-in-the-head (MPCitH) paradigm; this similarity to a signature scheme allows us to bind attribute data to the proof of possession, as required by certificate issuance protocols. We show how to instantiate this approach for two lattice-based KEMs in Round 3 of the NIST post-quantum cryptography standardization project, Kyber and FrodoKEM, and achieve reasonable proof sizes and performance. Our proofs of possession are faster and an order of magnitude smaller than the previous best MPCitH technique for knowledge of a lattice key, and in size-optimized cases can be comparable to even state-of-the-art direct lattice-based ZK proofs for Kyber. Our approach relies on a new result showing the uniqueness of Kyber and FrodoKEM secret keys, even if the requirement that all secret key components are small is partially relaxed, which may be of independent interest for improving efficiency of zero-knowledge proofs for other lattice-based statements.

CCS CONCEPTS

• Security and privacy → Key management; Public key (asymmetric) techniques.

KEYWORDS

public key infrastructure; certificates; key encapsulation mechanisms; proof of possession; zero knowledge proofs; post-quantum cryptography

ACM Reference Format:

Tim Güneysu, Philip Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. 2022. Proof-of-Possession for KEM Certificates using Verifiable Generation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*, November 7–11, 2022, Los Angeles, CA, USA. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3548606.3560560>

1 INTRODUCTION

Public key infrastructures (PKIs) operate by issuing certificates which bind an entity's public key to identifying information for that entity [2, 24]. During the enrollment process, the issuing certificate authority, before agreeing to bind a public / private keypair to an entity, often requires the requester to demonstrate use of their private key as a *proof of possession (PoP)*. The most common proof of possession mechanism is the PKCS#10 Certificate Signing Request (CSR) [59] which is used in many certificate enrollment protocols such as CMP [1], ACME [5], EST [61], and SCEP [41]. (See the full version [40] for a discussion on the role of PoPs in PKIs.)

CSRs have become ubiquitous in part because they are fully non-interactive and therefore easily portable. They do not require the certificate authority (CA) to have real-time communication with the entity requesting the certificate, so a CSR can be transported and validated out of band. As an example, web PKI CAs tend to offer simple certificate enrollment workflows requiring a certificate requester to paste a CSR into the CA's web page [5] first. This allows key generation and CSR creation to take place on a production server, after which the certificate request is initiated from a workstation outside the production network. Even fully automated certificate issuance protocols such as ACME [5] do not require the CSR to contain any protocol state information which would force it to be freshly-generated as part of the certificate issuance exchange.



This work is licensed under a Creative Commons Attribution International 4.0 License.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9450-5/22/11.
<https://doi.org/10.1145/3548606.3560560>

One major drawback of CSRs is that they require a digital signature [59], and therefore can only be used for PoP of digital signature-type keys. Some enrollment protocols such as CMP [1] define PoP mechanisms for key agreement and encryption keys, however these are interactive mechanisms requiring either a half or full round trip of communication between the CA and the certificate requester.

This problem has been side-stepped in many applications, such as the web PKI, which almost exclusively use certificates containing digital signature public keys. However, this may not necessarily be the case in the future, especially in the context of the transition to post-quantum cryptography. In 2020, Schwabe, Stebila, and Wiggers [65] proposed the KEMTLS protocol as a replacement for the TLS 1.3 handshake [63] which uses key encapsulation mechanisms (KEMs) for authentication, rather than signatures. Since post-quantum digital signatures tend to have larger communication sizes than post-quantum KEMs, KEMTLS can reduce the bandwidth needed to establish a post-quantum channel. But KEMTLS requires certificates containing KEM keys, so one cannot avoid the question of how to prove possession of a KEM key, preferably in a non-interactive setting to maintain the flexibility and existing workflows that utilize certificate signing requests.

Lattice-based KEMs are among the leading candidates in the NIST post-quantum cryptography standardization project, due to a relatively good mix of speed and communication size. Three of the five Round 3 finalists are lattice-based KEMs (Kyber, NTRU, and Saber), as well as one Round 3 alternate candidate (FrodoKEM). In this paper we focus on Kyber [16, 64] (based on the module learning with errors (MLWE) problem [48]) and FrodoKEM [17, 57] (based on the learning with errors (LWE) problem [62]); at a high level, both have a similar design, which we briefly outline here. Roughly speaking, in FrodoKEM the public key is a pair of matrices (\mathbf{A}, \mathbf{B}) where $\mathbf{B} = \mathbf{AS} + \mathbf{E} \bmod q$ for secret matrices \mathbf{S} and \mathbf{E} consisting of small entries. At the 128-bit security level, $q = 2^{15}$, “small” means in $[0, \pm 12]$, and \mathbf{S} and \mathbf{E} contain 10,240 entries together.

1.1 Existing Zero-Knowledge Techniques

The obvious approach for building a non-interactive proof of possession for a lattice-based KEM key would be to use zero-knowledge tools. There are two main techniques: directly building a zero-knowledge proof for the relevant mathematical operations, or using a generic technique such as SNARKs [13] or multi-party computation in-the-head (MPCitH) [43].

Direct lattice-based ZK constructions. There is a long line of research for ZK proofs of knowledge of a lattice secret key, starting with lattice-based identification schemes [52, 56]. Kawachi et al. [47] adapted techniques from Stern’s identification scheme [66] based on syndrome decoding to the lattice setting, which has influenced several subsequent constructions [49, 51]. Recent direct constructions focusing on proving knowledge of an LWE or short integer solution (SIS) sample include [11, 15, 34, 53–55], most of which focus on binary $\{0, 1\}$ or ternary $\{-1, 0, 1\}$ secrets. Some include ZK proofs of verifiable decryption for lattice-based KEMs [34, 53, 55]. The construction yielding the smallest proofs is a recent pre-print by Lyubashevsky et al. [53]: its proof takes place in a larger space (e.g., a Kyber512 key with modulus $q = 3329$ is embedded in a proof with modulus $\approx 2^{36}$) and depends on the hardness of both MSIS

and MLWE in the larger space, so parameters for the proof must be chosen appropriately to both ensure the proof can be constructed and that the MSIS and MLWE problems are hard in the larger space.

Generic ZK constructions. There are several constructions based on zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs). del Pina et al. [28] showed how to use Bulletproofs [19] to prove knowledge of lattice secret key with proofs as small as 1.25 kB, but at the cost of relying on a non-post-quantum security assumption and a 10 second generation time. Boschini et al. [18] showed how to use the Aurora SNARK system [9] (for rank-1 constraint systems) for module LWE/SIS relations, and Beullens [11] gave estimates for using both Aurora and Ligerio [3] (based on probabilistically checkable proofs) for LWE secrets; however Aurora’s runtimes are estimated to be dozens of seconds [18]. Baum and Nof [7] showed how to use MPCitH [43] with preprocessing [46] for ZK proof of knowledge of binary LWE secrets.

In terms of proof sizes (excluding non-post-quantum constructions [28]), among existing constructions, direct ZK constructions generally yield the smallest proof sizes, followed by SNARK systems, followed by MPCitH. Generally, the most relevant input features affecting performance are the modulus q , the total number σ of entries among the secrets \mathbf{S} and \mathbf{E} (whether they be matrix entries in a plain LWE instance, or coefficients in a ring-LWE instance, or both in a module-LWE instance), and the bounds β on the absolute value of the size of the secret entries. However, conducting an apples-to-apples comparison among the literature is challenging: it is rare that two papers in this field give example parameterizations that are directly comparable, they rarely provide direct non-asymptotic formulas or scripts for calculating parameter sizes, and very few report runtime results of implementations. Additionally, many constructions are principally designed for binary or ternary secrets; for some there are techniques that can extend to larger secrets bounds β by transforming the problem instance, but determining parameters for this extension is non-trivial. Table 1 presents results from the literature of various schemes in various parameter regimes.

1.2 Our Approach and Contributions

Our overall contribution is non-interactive protocols for proof of possession of lattice-based KEM keys (with instantiations for Kyber and FrodoKEM) using a technique called verifiable generation, in which the PoP is generated at the same time as the key itself.

Definitions for proof of possession. In Section 3, we begin by giving the syntax for a key generation and proof of possession (KGPOP) scheme, and consider a special class of *combined* KGPOP schemes in which the proof of possession is generated at the same time as the key. We define two security properties for such schemes: *unforgeability* and *zero-knowledge* of proofs of possession. Roughly speaking, unforgeability says that it is hard to generate a valid proof of possession for well-formed public key without the secret key (defined similarly to signature unforgeability), and zero knowledge says that proofs leak no information about the secret key (defined via simulatability in the random oracle model). However, since the key pair exists not just for the purposes of proof of possession, but has some intended application purpose (for example, authentication in KEMTLS), we have to consider composability of the PoP with

Table 1: Comparison of proof sizes and proof generation runtime for several parameter regimes

Scheme	Technique	Regime 1		Regime 2		Kyber512		Frodo640	
		Size	Size	Time	Size	Time	Size	Time	
<i>Proof of knowledge of secret key (and proof of verifiable decryption, denoted \diamond)</i>									
Stern-like [49]	ZKP from SIS	2.3 MB [†]	4.3 MB [†]						
[7]	MPCitH		4.1 MB	2.4 s				≥ 8.42 MB [‡]	
[15]	ZKP from RLWE & RSIS	384 kB [†]							
[11]	Σ -prot. for permuted-kernel	233 kB	444 kB						
Ligero [4]	zkSNARK from PCPs	157 kB [†]	200 kB [†]						
Aurora [9]	zkSNARK for R1CS	72 kB [†]	71 kB [†]						
[34]	ZKP from MLWE & MSIS	47 kB, 61 kB \diamond							
[54]	ZKP from MLWE & MSIS	47 kB*							
[55]	ZKP from MSIS & ext. MLWE	33 kB				43.6 kB \diamond			
[53]	ZKP from MLWE & MSIS	14 kB				19.0 kB \diamond			
<i>Proof of verifiable generation</i>									
Ours (31, 26)	MPCitH	251 kB	879 kB		52.9 kB	0.006 s	650 kB	0.12 s	
Ours (256, 16)	MPCitH	155 kB	542 kB		33.4 kB	0.028 s	402 kB	0.63 s	
Ours (1626, 12)	MPCitH	117 kB	407 kB		25.7 kB	0.109 s	302 kB	2.59 s	
Ours (65536, 8)	MPCitH	79 kB	272 kB		17.8 kB	3.77 s	203 kB	85.6 s	

Empty cell indicates estimates for parameter regime not available in the original paper or subsequent literature.

“Ours (N, τ)” denotes number N of MPC-in-the-head parties and number τ of MPC repetitions.

Regime 1: modulus $q \approx 2^{32}$, number of secret entries $\sigma = 2048$, ternary secrets $\{-1, 0, 1\}$, 128-bit security level.

Regime 2: modulus $q \approx 2^{61}$, number of secret entries $\sigma = 4096$, binary secrets $\{0, 1\}$, 128-bit security level.

Kyber512: modulus $q = 3329$, number of secret entries $\sigma = 1024$, secret $[0, \pm 2]$, 128-bit security level.

Frodo640: modulus $q = 2^{15}$, number of secret entries $\sigma = 10240$, secret $[0, \pm 12]$, 128-bit security level.

[†]: Estimates by Beullens [11]. [‡]: Estimate by us, using edaBits [33] for comparisons.

*: Estimate by Lyubashevsky et al. [55]. \diamond : For proof of verifiable decryption.

Runtime of “Ours” is a single-threaded implementation on Intel Core i7-8565U CPU running at up to 4.6 GHz, compiled with gcc 11.2.0.

the intended application purpose. In the context of an IND-CCA-secure KEM, in principle it is possible that (i) the PoP for a KEM key could undermine the confidentiality of the KEM shared secret, or conversely (ii) that use of the KEM secret key in decapsulation could undermine the unforgeability of proofs of possession. We handle this in both directions as follows. For (i), zero-knowledge naturally models the privacy notion we need. For (ii), we give the adversary access to an *auxiliary secret key usage algorithm* that represents use of the secret key in its intended application; proof of unforgeability must be shown even in the presence of this oracle. We give a quick example showing that certificate signing requests (CSRs) [59] are PoPs within our framework assuming the unforgeability of the signature scheme and appropriate domain separation between how CSRs are formatted and how application messages are formatted.

Non-interactive verifiable generation for lattice-based KEMs. Our main contribution, in Section 4, is a combined key generation and proof of possession scheme for lattice-based KEMs, based on the MPCitH paradigm, inspired in part by the Picnic signature scheme [21, 67]. The design of our scheme is as a 5-round interactive protocol to which the Fiat–Shamir transform [36] is applied to make a non-interactive scheme. Any additional attributes needed by the certificate issuance process can be incorporated into the Fiat–Shamir challenge generation, similar to messages in a signature scheme.

The MPC approach is compelling for our relation, $\mathbf{B} = \mathbf{AS} + \mathbf{E}$ for public \mathbf{A} , since it is entirely linear in secret information (\mathbf{S} and \mathbf{E}). Linear operations are done “locally” by the parties, and are

free in terms of communication. The main challenge is to ensure that the values in (\mathbf{S}, \mathbf{E}) are small – and this is where we benefit by combining generation of the key and PoP, since we can use a relatively simple cut-and-choose approach. The basic idea is as follows. First, we generate more small values than are needed for the secret key, and commit to all of them, then reveal some fraction of them to demonstrate that, with high probability, a sufficiently high number of the remaining values are small. Our analysis of this step uses a new result showing the uniqueness of Kyber and FrodoKEM secret keys: we can show that it is sufficient to prove that only a fraction of secret key components are small. Then the remaining unrevealed values will be used as the entries of the secret key from which we will construct the public key. These small values are the inputs to a simple N -party MPC protocol, which is repeated τ times to prove that \mathbf{B} is computed correctly.

Strictly speaking our proof does not guarantee that a lattice secret key is perfectly well-formed; a dishonest party can generate a key with non-small entries. But this approach suffices for the purposes of proof of possession: if Alice has generated a key honestly, then an adversary cannot generate a valid proof of possession for Alice’s public key without effectively recovering Alice’s secret key.

Outline of the security analysis. Using techniques developed for MPCitH-based signatures, we prove that the scheme is both zero-knowledge and straight-line extractable, also known in the literature as online extractable) in the ROM. Our extractor uses the

random oracle queries from a successful prover to recover a secret key that has mostly small entries and satisfies the equation $\mathbf{B} = \mathbf{A}\mathbf{S} + \mathbf{E}$. Together with Lemma 4 this immediately implies unforgeability, since in Lemma 4 we show that the extracted secret key is unique, even though the PoP may allow a bounded number of non-small entries. Using the explicit bound given by Lemma 4, we choose concrete parameters so that the probability of non-unique keys is negligible, and consequently the secret key extracted must with high probability be the original one, yielding unforgeability. (Lemma 4 allows relaxation without sacrificing uniqueness, and may help efficiency of zero-knowledge proofs for other lattice-based statements; for example the performance of [7] may be improved by checking that only a certain fraction of values are small.)

For our KEMTLS scenario we must handle unforgeability in the presence of the auxiliary secret key usage oracle corresponding to KEM decapsulation. We observe that IND-CCA-secure KEMs constructed via the Fujisaki–Okamoto transform [37, 42] (the class that includes FrodoKEM and Kyber) have a property we call *decapsulation simulatability*, meaning that in the ROM decapsulation can be simulated without the secret key; details appear in the full version [40]. This allows us to simulate the auxiliary secret key usage algorithm in the ROM during the proof unforgeability experiment. Conversely, since our proof is zero-knowledge, security of the KEM is maintained when it is added to the public key.

Overall, security only relies on the random oracle model plus a hiding commitment scheme and a pseudorandom generator, assumptions that are already required by FrodoKEM and Kyber and which follow from a random oracle. Compared with direct ZK constructions based on lattice assumptions, such as the state-of-the-art [53], one major benefit of our approach is that we do not rely on any significant security assumptions beyond the original scheme (e.g., [53] depends on both MLWE and MSIS). Moreover, direct lattice-based assumptions such as [53] have to select security parameters for the lattice problem on which the proof is based, which in general will be different from (and larger than) the security parameters for the lattice-based key the proof is about. Given the subtleties and expertise required to pick lattice parameters, it is an advantage of our scheme that the proof system’s parameters are based on MPCitH, which is quite a bit simpler to pick parameters for.

Generality of our construction Our exposition in Section 4 is phrased in terms of FrodoKEM, but most of it applies directly to Kyber as well, and we give a version of the uniqueness lemma (Lemma 4) for Kyber in Section 5. Our approach is flexible and we expect variants of our design would apply to other LWE, Ring-LWE or MLWE-based KEMs such as [60].

Implementation. In Section 6 we report on a software implementation of our scheme and results on runtime and proof size, for both Kyber and FrodoKEM at the 128-, 192-, and 256-bit security levels. We are able to tune the trade-off between proof size and runtime by varying the number of simulated MPC parties N and the number of MPC repetitions τ ; the range of trade-offs can be seen in Figure 5.¹

Overall, our MPCitH-based approach for verifiable generation yields proofs that are at least an order of magnitude smaller than the

previous best MPCitH approach for proof of knowledge of lattice secrets by Baum and Nof [7]. Compared to direct lattice constructions and post-quantum zkSNARKs, our MPCitH approach remains generally competitive, and our most sized-optimized parameterizations can in some cases nearly match the state-of-the-art direct lattice constructions. We summarize some comparative results below; see Table 1 for detailed comparisons across several parameter regimes. As noted above, direct comparisons with proof of knowledge schemes in the literature can be challenging due to difference in parameterizations and the non-triviality of extrapolating existing results to other parameter regimes, especially for non-ternary secrets. Nonetheless we are able to offer some direct comparisons.

Comparison to MPCitH. For the parameter set given in [7] targeted towards somewhat homomorphic encryption applications ($q \approx 2^{61}$, $\sigma = 4096$, binary secrets), our system yields proof sizes for the same parameters at sizes as small as 272 kB and of 542 kB for a reasonable (N, τ) trade-off, which is 7–15× smaller than Baum and Nof’s solution with a 4.0 MB proof and 2.4 second generation time. To apply Baum and Nof’s protocol to FrodoKEM parameters, one has change from handling binary secrets to small (absolute value $\leq \beta = 12$). Applying the edaBits construction [33] for integer comparisons in MPC, we extrapolate that checking Frodo640’s $\sigma = 10240$ secret values have absolute value at most $\beta = 12$ would lead to proof of at least 8.42 MB (at just 80 bits of security) with estimated multi-second runtimes. Our verifiable generation construction yields Frodo640 proofs of possession at sizes as small as 203 kB; as shown in Figure 5, other points in the size–speed tradeoff space include 402 kB in 0.63 seconds and 799 kB in 0.07 seconds, which is 11–41× smaller than the Baum and Nof proof.

Comparison to direct lattice constructions.

The most extensive results available for comparison across multiple papers are for a parameter suite with modulus $q \approx 2^{32}$, $\sigma = 2048$ samples, and ternary secrets. zkSNARKs achieve proof sizes of 72 kB (Aurora [9]) or 157 kB (Ligero [3]). Recent ZK proofs based on MLWE and MSIS achieve proof sizes of 47 kB [34], 33 kB [55], and 14 kB [53]. Our MPCitH-based verifiable generation construction yields PoPs for this parameter regime at sizes as small as 79 kB and at 156 kB for a reasonable (N, τ) trade-off, which is comparable with zkSNARKs but not as good as MLWE- and MSIS-based ZK proofs.

For the Kyber512 parameters ($q = 3329$, $\sigma = 1024$ samples, secrets $\leq \beta = 2$) our scheme compares more favorably with direct lattice constructions since it scales better with increased secret bound β . Only two papers in the literature report proof sizes for Kyber512 parameters [53, 55], and even then they only report sizes for proofs of *verifiable decryption*, rather than proof of knowledge of the secret key. Verifiable decryption proofs are a bit larger than proof of secret key knowledge; although size differences are not given in [53, 55], in [34] verifiable decryption proofs were about 1.3× bigger. [55]’s proof size for Kyber512 verifiable decryption is 43.6 kB, and for [53] it is 19.0 kB. Our MPCitH-based verifiable generation construction yields Kyber512 proofs of possession as small as 17.8 kB (runtime 3.77 seconds); as shown in Figure 5, other points in the size–speed tradeoff space include 25.7 kB in 109 milliseconds, 33.4 kB in 28 milliseconds, and 52.9 kB in 6 milliseconds.

¹Our C implementation of FrodoKEM and Kyber verifiable generation (with AVX2 optimizations for Kyber) can be downloaded at <https://github.com/Chair-for-Security-Engineering/KEM-NIZKPoP>, and is accompanied by scripts for calculating proof sizes.

2 PRELIMINARIES

All adversaries are assumed to be probabilistic polynomial time algorithms (in our security parameter κ), that are stateful, i.e. if \mathcal{A} appears twice in an security experiment, state may be shared between instances.

A *key encapsulation mechanism* (KEM) is a triple of algorithms $\text{KEM} = (\text{KeyGen}, \text{Encaps}, \text{Decaps})$ used to establish randomly chosen keys. The probabilistic algorithm $\text{KeyGen}(1^\kappa)$ outputs a key pair (pk, sk) . The probabilistic algorithm $\text{Encaps}(pk)$ outputs a ciphertext and key (c, K) , and $\text{Decaps}(sk, c)$ outputs K or \perp if c is not a valid encapsulation. A KEM with *implicit rejection* does not output \perp when c is invalid, but instead outputs a random K .

FrodoKEM [17, 57] is a key encapsulation mechanism based on the (plain) learning-with-errors problem [62], built on a design by Lindner and Peikert [50] and using a variant of the Fujisaki–Okamoto (FO) transform [37, 42, 44] to achieve indistinguishability under chosen ciphertext attacks (IND-CCA).

For our purposes, it suffices to examine only the key generation algorithm. The key generation algorithm of FrodoKEM samples a public seed seed_A used to pseudorandomly generate a matrix $A \in \mathbb{Z}_q^{n \times n}$. The algorithm also samples a secret seed seed_S used to pseudorandomly generate two secret $n \times \bar{n}$ matrices S and E containing entries sampled from a distribution χ . The public key is the seed seed_A and the matrix $B = AS + E$; the secret key is S . The distribution χ is small relative to the modulus q , and we will sometimes refer to *small values* as a concise way to distinguish between integers sampled from χ and elements of \mathbb{Z}_q . As a concrete example for FrodoKEM at the 128-bit security level, $q = 2^{15}$ and $\beta = 12$, so χ yields samples in $[0, \pm 12]$.

Kyber [16, 64] is a KEM based on the module LWE problem [48]. Very roughly speaking, Kyber can be thought of as similar to FrodoKEM, but where entries in the matrices A, B, S, E are polynomials in a ring $R_q = \mathbb{Z}_q[X]/(X^d + 1)$; see the Kyber specification for a more precise formulation [64]. As a concrete example of parameters for Kyber at the 128-bit security level, $q = 3329$, $d = 256$, $n = 2$, and χ yields samples in $[0, \pm 2]$.

MPC-in-the-head. The MPC-in-the-head (MPCitH) paradigm [43] connects two fundamental primitives in cryptography: secure multiparty computation (MPC) and zero-knowledge proofs, showing that an MPC protocol for a functionality f can be used to construct a ZK proof of x such that $y = f(x)$. Since MPC protocols exist (or can be designed) for most f , we can create ZK proofs for arbitrary circuits. Furthermore, this was shown to be efficient in [38], inspiring many constructions of signature schemes [6, 11, 12, 21, 26, 27, 29, 35, 45, 46, 67], as they need only assume that f is a one-way function (OWF) in the random oracle model, they potentially have post-quantum security. [31] show conditions under which Fiat–Shamir-based signature schemes have security in the quantum random oracle model.

We give a high level explanation of MPCitH, somewhat specialized to our use, then give additional details in Section 4 and refer to the references given above for more details. We create an interactive proof protocol that $f(x) = y$ (for public f, y) between a prover P (with secret input x) and verifier V . P starts by additively secret sharing x among the N parties, as input to the MPC. Then P simulates the execution of the MPC protocol; parties record the

messages they exchange (either directly or via broadcast), along with their input share to form their *view* of the protocol execution. The state of the computation remains secret shared until the last step, when the parties broadcast their shares of y so that it may be reconstructed. P then commits to all N views, and sends the commitment to V , who responds with a random subset of the N parties (of size $N - 1$). For these parties the prover will reveal their views – we refer to these parties as *opened*, and the remaining party as *unopened*. Given the input shares of the opened parties, and the messages sent by the unopened party, V can recompute all $N - 1$ views and check that they match those in P 's commitment. The verifier accepts the proof if the output is y and the views match. After one repetition of this process, V is convinced that P knows x with probability $1/N$. As mentioned above, the MPC protocol for our f is very simple as the parties can add secret-shared values *locally*, i.e., without communication, and can also multiply secret-shared values by a public constant locally. Intuitively, the protocol is sound because V recomputes the circuit, and it is zero-knowledge if the protocol is $(N - 1)$ -private, meaning that no information is leaked about x given the state of $N - 1$ parties.

Our protocol of Section 4 is a five-round protocol, where the first exchange acts as a setup phase (not to be confused with a preprocessing phase, as in [46]), that establishes the inputs to the MPC, and the rest of the protocol follows the sketch given above.

3 DEFINING ZERO-KNOWLEDGE PROOFS OF POSSESSION

In this section, we define the syntax and security properties for non-interactive proof of possession, as well as give an example of how Certificate Signing Requests fit our framework.

Definition 1 (Key generation and proof of possession scheme). A *key generation and proof of possession (KGPOP) scheme* consists of:

- $\text{PoP.KG}() \mapsto (pk, sk)$: A probabilistic key generation algorithm that outputs a public, secret key / secret key pair.
- $\text{PoP.PG}(pk, sk, \text{attrs}) \mapsto \pi$: A probabilistic proof generation algorithm that takes as input a public key, secret key, and attributes $\text{attrs} \in \{0, 1\}^*$, and outputs a proof string π . (In our application, attrs could be the body of a certificate signing request, for example.)
- $\text{PoP.Vf}(pk, \text{attrs}, \pi) \rightarrow \{0, 1\}$: A deterministic verification algorithm that takes as input a public key pk , attributes attrs , and proof π , and outputs 1 if the proof is valid, and 0 otherwise.

A more general formalization could be made with a setup procedure for a common reference string (CRS); since our constructions will not rely on a CRS, we opt to omit the generalization.

A key generation and proof of possession scheme should satisfy the obvious correctness property: for all attrs ,

$$\Pr \left[\text{PoP.Vf}(pk, \text{attrs}, \pi) \Rightarrow 1 \mid \begin{array}{l} (pk, sk) \leftarrow \text{PoP.KG}(); \\ \pi \leftarrow \text{PoP.PG}(pk, sk, \text{attrs}) \end{array} \right] = 1.$$

In our approach for proof-of-possession via verifiable generation, the key pair and the proof are generated at the same time. We call such a scheme a *combined* key generation and proof of possession scheme, and in this case the two algorithms PoP.KG and PoP.PG

from Definition 1 (which we sometimes refer to as a “separable” scheme) are replaced with a single algorithm:

- $\text{cPoP.KPG}(\text{attrs}) \xrightarrow{\$} (\text{pk}, \text{sk}, \pi)$: A probabilistic key and proof generation algorithm that takes as input attributes $\text{attrs} \in \{0, 1\}^*$, and outputs a public key pk , secret key sk , and proof string π .

One could build the combined cPoP.KPG in the obvious way from separable KG and PG : generate the key pair $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{PoP.KG}()$, then the proof $\pi \xleftarrow{\$} \text{PoP.PG}(\text{pk}, \text{sk}, \text{id})$, then return $(\text{pk}, \text{sk}, \pi)$. The corresponding correctness property applies.

Admittedly, combining proof generation with key generation limits the number of proofs generated, which may be unsatisfactory for some applications. For certificate issuance, a single proof suffices, since only a single certificate signing request is needed. (In practice, even if multiple certificates are requested over time for the same key, the same certificate signing request is typically used.) Furthermore, other certificate lifecycle events (such as revocation) can be handled by binding (into the proof attributes) an additional account key that authenticates other lifecycle events; see discussion in Section 7.

The main security property we want of a key generation and proof of possession scheme is *unforgeable proofs of possession*: it should be hard for an adversary to construct a valid proof π for a public key without the corresponding secret key. However, we also need assurance that the PoP can be used safely in conjunction with the actual cryptographic primitive the corresponding keys are meant for. In particular, we have to ensure (a) that the PoP functionality does not undermine the security of the actual cryptographic primitive, and (b) use of the secret key in the actual cryptographic primitive does not undermine security of the PoP . In our case, the former will follow when the PoP is non-interactive zero knowledge; the latter will be shown by incorporating (constrained) use of the secret key into the unforgeability definition for PoPs . (See [23] for additional considerations on composition of PKIs with protocols.)

We state the formal definition of security for combined proofs of possession now, then present the details and rationale in Sections 3.1 and 3.2, and give an example of how these definitions apply to certificate signing requests in Section 3.4.

Definition 2 (Security of Combined PoPs). Let $\text{cPoP} = (\text{KPG}, \text{Vf})$ be a combined key generation and proof of possession scheme. We say that cPoP is a *secure cPoP scheme* if it is:

- *Zero-knowledge*: no efficient adversary exists for the security experiment $\text{Exp}_{\text{PoP}}^{\text{ZK}}$ (as shown in Figure 2) as defined in Definition 3.
- *Unforgeable*: no efficient adversary exists for the security experiment $\text{Exp}_{\text{cPoP}, \text{Aux}}^{\text{UF}}$ (as shown in Figure 1), for the required auxiliary secret key usage algorithm Aux .
- *Correct*: key pairs output by cPoP.KPG are distributed identically to the KeyGen function of the primitive that defines the key pair.

To highlight the difference between proof of possession and proof of knowledge or well-formedness: a proof of possession does not guarantee that a public key was generated honestly by someone who knows the secret key, but it does guarantee that an honestly generated public key cannot be claimed by someone else who does not know the secret key. In the context of PKI, a proof of possession

$\text{Exp}_{\text{PoP}, \text{Aux}}^{\text{UF}}(\mathcal{A})$	$\text{Exp}_{\text{cPoP}, \text{Aux}}^{\text{UF}}(\mathcal{A})$
1 : $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KG}()$	1 : $\text{attrs} \xleftarrow{\$} \mathcal{A}()$
2 : $(\text{attrs}', \pi') \xleftarrow{\$}$ $\mathcal{A}^{\text{PG}(\text{pk}, \text{sk}, \cdot), \text{Aux}(\text{sk}, \cdot)}(\text{pk})$	2 : $(\text{pk}, \text{sk}, \pi) \xleftarrow{\$} \text{KPG}(\text{attrs})$
3 : return $\text{Vf}(\text{pk}, \text{attrs}', \pi')$	3 : $(\text{attrs}', \pi') \xleftarrow{\$} \mathcal{A}^{\text{Aux}(\text{sk}, \cdot)}(\text{pk}, \pi)$
\wedge (attrs' not queried to PG)	4 : return $\text{Vf}(\text{pk}, \text{id}', \pi')$
	\wedge ($\text{attrs}' \neq \text{attrs}'$)

Figure 1: Security experiment for proof unforgeability of a key generation and proof of possession scheme $\text{PoP} = (\text{KG}, \text{PG}, \text{Vf})$ (left) and a combined key generation and proof of possession scheme $\text{cPoP} = (\text{KPG}, \text{Vf})$ (right), with respect to auxiliary secret key usage algorithm Aux .

does not protect against a malicious party getting a certificate for a possibly malformed public key, but does protect against a malicious party getting a certificate for someone else’s public key. This modelling also allows us to cover certificate signing requests, which are accepted in practice for demonstrating proof of possession but are not in general a proof of knowledge or well-formedness.

3.1 Unforgeability

The unforgeability security experiment for a KGPOP scheme, shown in Figure 1, is defined with respect to an *auxiliary secret key usage algorithm* Aux , which models any usage of the secret key in a subsequent application:

- $\text{Aux}(\text{sk}, x) \xrightarrow{\$} y$: A probabilistic or deterministic auxiliary secret key usage algorithm that takes as input a secret key sk and input x , and produces an output y .

For example, in a PoP for a KEM key, Aux would correspond to decapsulation; in a PoP for a signing key, Aux would correspond to signature generation.

The unforgeability experiment $\text{Exp}_{\text{PoP}, \text{Aux}}^{\text{UF}}$ in Figure 1 is analogous to weak existential unforgeability of a signature scheme under chosen message attacks. A version analogous to strong existential unforgeability could be had by checking that (attrs', π') were not the input/output of a query to PG . (While our verifiable generation construction for FrodoKEM and Kyber does satisfy the stronger notion, we include the weak unforgeability notion to accommodate proof-of-possession schemes such as certificate signing requests built from weakly unforgeable signature schemes such as ECDSA.)

For combined KGPOP schemes, we need a slightly different definition of proof unforgeability: since the proof generation is not separate from the key generation, the adversary cannot repeatedly obtain PoPs . The resulting simplified experiment is shown in Figure 1. It is analogous to weak existential unforgeability of a signature scheme under a key-only attack; as above, a strong existential unforgeability version can be had.

When a combined KGPOP scheme is constructed in the obvious way from a separable KGPOP scheme, security in the sense of the former immediately implies security in the sense of the latter.

3.2 Non-Interactive Zero Knowledge

Having defined security such that use of the key in an application does not affect security the PoP scheme, we now consider the other

direction, and look at security of the application against attackers that have proofs created by the PoP scheme.

There certainly exist degenerate schemes where the proof of possession could undermine the intended application. Consider for example EdDSA [10] with the following proof of possession. Let (pk, sk) be an ECC key pair. Let the PoP be (σ, m) where σ is an ECDSA signature on $m = \text{SHA512}(sk) \parallel \text{attrs}$. Proof verification checks the ECDSA signature on the provided m as usual, without checking whether the first part of the message, $\text{SHA512}(sk)$, has any connection to the public key; this is still a good proof of possession under the assumption that ECDSA is unforgeable. The intended use of the key is EdDSA signatures, for TLS authentication or any other purpose. The PoP is clearly unforgeable under weak unforgeability of ECDSA, but use of the key in EdDSA is completely broken since EdDSA uses $\text{SHA512}(sk)$ as a PRF key to derive per-signature nonces. In isolation the two primitives are secure, but the composition is insecure: the PoP leaks information, namely $\text{SHA512}(sk)$, that compromises the application.

The natural way to protect against such attacks is to require PoP schemes to be zero-knowledge: they should reveal nothing more than the fact that the party creating the proof knows sk associated to a given pk . Our definition is nearly the same as honest verifier zero-knowledge (see e.g. [14, §19.1.1]) except that (i) we relax the definition to statistical zero-knowledge [39, Def. 4.3.4] and (ii) we have the attrs string that is allowed to be chosen arbitrarily, and is bound to the proof. Our definition also assumes the PoP is non-interactive, as motivated by our use case, so we note that the simulator will exist in an idealized model where Sim has some additional capability, such as the ability to choose system parameters, or simulate hash functions in the random oracle model (ROM). Otherwise, if Sim was efficiently implementable by any party the ZK property would exclude the unforgeability property.

Definition 3. Let $\text{PoP} = (\text{KG}, \text{PG}, \text{Vf})$ be a key generation and proof of possession scheme. We say that PoP is *zero-knowledge*, if there exists an efficient probabilistic algorithm Sim , such that, for all possible outputs (pk, sk) of KG , the output distribution of $\text{Sim}(pk, \text{attrs})$ is statistically close to the output distribution of $\text{PG}(pk, sk, \text{attrs})$. More formally, PoP is zero-knowledge if no efficient adversary exists for $\text{Exp}_{\text{PoP}, \text{Sim}}^{\text{ZK}}$ (as shown in Figure 2 (left)) with success probability different from $1/2$. For combined KGPOP schemes, the corresponding experiment is shown in Figure 2 (right).

3.3 Handling Aux Queries for KEMs

Our unforgeability definition in Section 3.1 takes care to model use of the secret key in an application, so that this use does not allow an attacker to create a proof of possession for a user's key after they use it. This is very broad and in general difficult to handle (as illustrated in the CSR example of Section 3.4). Fortunately for our KEMTLS scenario, we can show that unforgeability can be preserved when the KEM key is in use.

Full details of our approach are in the full version [40]; we give the intuition here. For a KEM, we define the security notion of *decapsulation simulatability*. The decapsulation operation of a KEM is said to be simulatable if there exists a simulator Sim which takes a public key pk and ciphertext as input and has outputs that are indistinguishable from KEM.Decaps . From this, our security proof

$\text{Exp}_{\text{PoP}, \text{Sim}}^{\text{ZK}}(\mathcal{A})$	$\text{Exp}_{\text{cPoP}, \text{Sim}}^{\text{ZK}}(\mathcal{A})$
1: $(pk, sk) \leftarrow \mathcal{A}()$	1: $\text{attrs} \leftarrow \mathcal{A}()$
2: $b \leftarrow \{0, 1\}$	2: $(pk, sk, \pi_0) \leftarrow \text{KPG}(\text{attrs})$
3: if $b = 0$ then $b' \leftarrow \mathcal{A}^{\text{PG}(pk, sk, \cdot)}()$	3: $\pi_1 \leftarrow \text{Sim}(pk, \text{attrs})$
4: if $b = 1$ then $b' \leftarrow \mathcal{A}^{\text{Sim}(pk, \cdot)}()$	4: $b \leftarrow \{0, 1\}$
5: return $\llbracket b = b' \rrbracket$	5: $b' \leftarrow \mathcal{A}(\pi_b)$
	6: return $\llbracket b = b' \rrbracket$

Figure 2: Security experiment for zero-knowledge of proofs of possession for a key generation scheme PoP (left) and a combined key generation and proof of possession scheme cPoP (right) with respect to simulator Sim .

for unforgeability (Theorem 7) can simulate use of the KEM key (in the ROM) and answer Aux queries without using sk . We prove that KEMs constructed with the variant of the Fujisaki–Okamoto (FO) transform used in FrodoKEM and Kyber have this property. Our proof is based on a result of [42] that implicitly defines the required simulator when proving that FO transform provides CCA security.

3.4 Example: Certificate Signing Requests

To build familiarity with these notions, we observe how a familiar PoP system for signature schemes, certificate signing requests (CSRs), can be modeled in this framework. CSRs are an example of a (separable) KGPOP scheme. Let $\Sigma = (\Sigma.\text{KG}, \Sigma.\text{Sign}, \Sigma.\text{Vf})$ be a signature scheme. Define the KGPOP scheme $\text{CSR}[\Sigma] = (\text{KG}, \text{PG}, \text{Vf})$ with helper function CSRfmt as follows:

- $\text{KG}()$: Return $\Sigma.\text{KG}()$.
- $\text{PG}(pk, sk, \text{attrs})$: Return $\Sigma.\text{Sign}(sk, \text{CSRfmt}(pk, \text{attrs}))$.
- $\text{Vf}(pk, \text{attrs}, \pi)$: Return $\Sigma.\text{Vf}(pk, \text{CSRfmt}(pk, \text{attrs}), \pi)$.
- $\text{CSRfmt}(pk, \text{attrs}) \rightarrow \{0, 1\}^*$: A deterministic CSR formatting function that generates the body of a certificate signing request from a public key pk and attributes attrs , for example according to a standards document such as RFC 2986 [59]. We assume CSRfmt is collision-free.

It is straightforward to see that if Σ has existential unforgeability under chosen message attack, then it is hard to forge proofs of possession in $\text{CSR}[\Sigma]$ – after all, PoPs are just signatures.

However, if we want to model that the secret key generated by $\text{CSR}[\Sigma]$ will then be used as a signing key in some subsequent application, we have to make use of an auxiliary secret key usage algorithm. If the subsequent application (and hence the auxiliary secret key usage algorithm) allows an adversary to sign arbitrary messages, then the adversary could just ask it to sign a well-formatted CSR body. To prevent this, we must enforce some kind of domain separation. We model this by introducing a function $\text{IsCSR} : \{0, 1\}^* \rightarrow \{0, 1\}$ that checks if a candidate message is a valid formatted CSR body, and refuses to sign if it is: $\text{Aux}(sk, x)$ returns \perp if $\text{IsCSR}(x) = 1$, else it returns $\Sigma.\text{Sign}(sk, x)$. Assuming that $\text{IsCSR}(x) = 1$ for all $x \leftarrow \text{CSRfmt}(pk, \text{attrs})$, it is then straightforward to prove, via reduction, that $\text{CSR}[\Sigma]$ has unforgeable proofs of possession (with respect to the given Aux) in the sense of Figure 1, assuming Σ is (weakly) existentially unforgeable under chosen message attack.

The domain separation between CSRs and application messages can be easy to satisfy in practice, though it still must be verified for each application. For example, consider digital signatures in the Web PKI: a server gets an X.509 certificate for its signing key by creating a CSR as per [59], and subsequently uses the signing key for authentication in the TLS 1.3 protocol [63]. In this case, a CSR body is a certain ASN.1 data structure, with various fields and formatting as specified in [59]; most notably, a CSR body is always a ASN.1 SEQUENCE object, which in BER or DER encoding means that the first byte will be hexadecimal $0x10$ or hexadecimal $0x30$. In TLS 1.3, signatures are generated over data structures with a different format as specified in [63, §4.4.3], specifically where the first byte will be hexadecimal $0x20$. Assuming a server only uses its signing key in TLS, this immediately yields domain separation that can be modeled via CSRfmt and lsCSR.

4 NON-INTERACTIVE VERIFIABLE GENERATION FOR LATTICE-BASED KEMS

In this section we give our construction of a non-interactive, combined generation and proof of possession for lattice-based KEMs. We will focus on FrodoKEM as it is simpler to describe, but the approach and analysis also works for Kyber and other similar schemes. We start this section with a technical overview of our construction, then give full details of the construction for FrodoKEM, discuss parameter selection and provide a security analysis.

4.1 Technical Overview

Consider proving knowledge of a pair of secret matrices S, E whose entries are small and satisfy $B = AS + E \pmod q$, where A and B are public matrices. Let P be the prover and V be the verifier. Let σ be the total number of entries in the secret matrices S and E , integers sampled from a “small” interval according to a distribution χ . We describe the protocol as an interactive 5-round protocol between P and V , as an MPCitH proof with N parties.

- (1) **Commit to seeds and bundles.** P generates $M > \sigma$ random small values, where M is a parameter. For each of the M values, P generates an N -party additive secret sharing (mod q). P commits to the shares and sends the commitment to V .
- (2) **Challenge a subset of the bundles to audit.** V chooses $M - \sigma$ of the small values to audit and sends this as a challenge to P . The idea is that P has committed to more than σ values, and V will check a subset of them, to make sure they belong to the correct distribution.
- (3) **Generate keypair, simulate MPC, commit to views, and open the audited bundles.** P responds with the opening of the commitment for each of the audited values. P uses the unaudited values to construct the secret key (S, E) . Note that the N parties have shares of (S, E) , and P has committed to the shares. P randomly generates a matrix A and computes the public key $B = AS + E$. Given A , the parties can compute shares of B using A and the sharings of S and E (no communication is required among the parties as this is a linear operation). Finally, P commits to the shares of B (the views of the parties), and sends (A, B) to V .

- (4) **Challenge the MPC simulation.** V selects $N - 1$ parties to audit. Here the verifier’s goal is to check that the MPC was executed correctly, namely that B was computed honestly.
- (5) **Open the MPC views and output the key pair and proof.** P reveals the state of the $N - 1$ audited parties and outputs the keypair computed in step 3.
- (6) **Verify final response.** V recomputes the $N - 1$ views and ensures they match the views committed to by P . V also checks that the $M - \sigma$ opened values meet the range criteria given by χ . If both checks pass, V outputs accept; otherwise V rejects.

We will show that for an appropriate choice of M (relative to σ and the lattice parameters), V will be assured that the σ values in (S, E) are from χ , except with probability $2^{-\kappa}$. However, the second challenge has only soundness $1/N$ and necessitates repeating the protocol τ times in parallel. These parallel repetitions are not independent as they all use the same M values sampled from χ (with independent sharings in each repetition).

To make this protocol into a non-interactive proof, V ’s challenges are derived with a hash function (in the random oracle model (ROM) [8]), using the well-known Fiat–Shamir transform [36]. Going forward we will focus on the non-interactive case, since this is required by our proof-of-possession scenario.

When viewed as a proof of knowledge, combined KGPOP does not quite fit the established structure. The problem instance, or statement x (corresponding to matrices A and B) and witness w (corresponding to secrets S and E) are negotiated in the first part of the protocol, rather than fixed in advance. However, because of the protocol’s commit-and-open structure, in the ROM we can still extract P ’s input including w (and we prove this in the full version [40]). We can also simulate transcripts in the ROM for any x , to show (in the full version) that the protocol is computational honest-verifier zero-knowledge so that proofs do not leak information about the secret key to computationally bounded adversaries.

4.2 FrodoKEM KGPOP construction

We now provide a full description of the non-interactive verifiable generation protocol for FrodoKEM. The combined key generation and proof of possession protocol (cPoP.KPG) is given in Figure 3, and the verification operation (cPoP.Vf) is given in Figure 4.

Notation. Recall that q is the modulus used in FrodoKEM computations, τ is the number of parallel repetitions, and N is the number of parties used in the MPC protocol. Let σ denote the total number of small integers required for FrodoKEM private values. A *bundle* B_σ is a collection of τ sharings of a value σ between N parties in the ring \mathbb{Z}_q . χ is the distribution used for Frodo.Gensecrets and errors. The input to KPG is $\text{attrs} \in \{0, 1\}^*$, the string encoding the attributes to be bound to the cPoP. We use $x^{(i)}$ to denote party P_i ’s share of x , and $[X]$ to denote the set $\{1, \dots, X\}$. In the MPC protocols simulated by the prover, each party can sample their share of a value x from their random tape. This is sufficient when x must be a uniform random value. To share a given value, P provides a “delta value” computed as: $\Delta x = x - \sum_{i=1}^N x^{(i)}$. P makes Δx is public, and it is added to P_1 ’s share to correct the sharing: $x^{(1)} = x^{(1)} + \Delta x$.

As we will be working with an $N - 1$ private MPC protocol with τ parallel repetitions, in repetition e , there is one party that is unopened, we denote their index by i_e^* , or simply i^* when the repetition index is clear from context.

Helper functions. We use the following helper functions to simplify our presentation.

Arrange creates (S, E) from a list of σ values in a canonical way. Given a subset C of $\{1, \dots, M\}$ of size σ , and values $\{v_1, \dots, v_M\}$, let $(S, E) = \text{Arrange}(\{v_1, \dots, v_M\}, C)$ be the two matrices given by populating the entries of S and E from the elements v_i for $i \in C$, starting in the top-left of S and proceeding row-by-row from left to right, and then continuing with E .

Expand is a function that takes a hash digest, and expands it into a challenge of the correct form. The function $\text{ExpandTape}()$ takes a seed and expands it into a random tape. These functions are implemented with an extendable output hash function, such as SHAKE [58]. The function $\text{Sample}()$ reads random values from a random tape, converting the random bits to the required type, and keeping a pointer to the next bits to be read.

Optimizations. We use an optimization from [46] that is now standard in the MPCitH literature: P derives per-party seeds with a binary tree construction, so that $N - 1$ seeds can be communicated to V with $\lceil \log_2 N \rceil$ seeds. We use another optimization at a couple points of the protocol. If x is a public value, and the prover has committed to shares of it, e.g., $h = H(x^{(1)}, \dots, x^{(N)})$, the verifier may recompute h from x and $N - 1$ of the shares as follows $x^{(i^*)} = x - \sum_{i \neq i^*}^N x^{(i)}$, since by definition $x = \sum_{i=1}^N x^{(i)}$. This saves the prover from having to communicate the unopened party's share. For example, in the bundles we audit, we commit to τ sharings of v_k : by revealing v_k , the verifier can recompute the missing shares and check the commitments.

Proof Size. The size of the proof π output by Figure 3 is

$$6\kappa + \tau \cdot (2\kappa + \kappa \lceil \log_2(N) \rceil + \ell_q M) + \ell_\chi (M - \sigma) \quad (1)$$

bits, where the term 6κ accounts for the salt and two challenges (each 2κ bits long), 2κ is the size of the per-party commitment, $\kappa \lceil \log_2(N) \rceil$ is the number of bits required to communicate the seeds of the unopened parties in each repetition, ℓ_q is $\lceil \log_2(q) \rceil$, and ℓ_χ is the number of bits required to represent a “small” value (sampled from the distribution χ). In Section 4.3.1 we explain how to choose the parameters (N, τ, M, σ) , and in Section 6 we give sizes and benchmarks from our implementation.

4.3 Unique Secret Keys and Parameter Selection

We now explain how to choose parameters for our cPoP to provide κ -bit security. There are two places where a prover can cheat: by choosing values for S and E that do not have the correct distribution; or in the MPCitH executions. The first part of parameter selection is how to choose the number of bundles to audit in the first step of the protocol, and is new to our work. The second part addresses the choice of the MPCitH parameters, which is similar to many other MPCitH proof protocols in the literature.

First note that with the cut-and-choose mechanism used to ensure that values are small, a malicious prover can create a proof where one of the values is *not small* with high probability, unless M

cPoP.KPG(attrs):

Phase 1: Commit to seeds and bundles

- 1: Sample a random salt: $\text{salt} \leftarrow \{0, 1\}^{2\kappa}$
- 2: Sample v_k for $k \in [M]$ from the distribution χ as in FrodoKEM
- 3: **for** each parallel repetition e **do**
- 4: Sample a root seed: $\text{seed}_e \leftarrow \{0, 1\}^\kappa$
- 5: Derive $\text{seed}_e^{(1)}, \dots, \text{seed}_e^{(N)}$ from seed_e as leaves of a binary tree
- 6: **for** each party i **do**
- 7: Commit to seed: $\text{com}_e^{(i)} \leftarrow \text{Commit}(\text{salt}, e, i, \text{seed}_e^{(i)})$
- 8: Expand tape: $\text{tape}_e^{(i)} \leftarrow \text{ExpandTape}(\text{salt}, e, i, \text{seed}_e^{(i)})$
- 9: Compute shares: $(b_{1,e}^{(i)}, \dots, b_{M,e}^{(i)}) \leftarrow \text{Sample}(\text{tape}_e^{(i)})$
- 10: **for** each $k \in [M]$ **do**
- 11: Compute the offset: $\Delta b_{k,e} \leftarrow v_k - \sum_{i=1}^N b_{k,e}^{(i)} \bmod q$
- 12: Update first party's share: $b_{k,e}^{(1)} \leftarrow b_{k,e}^{(1)} + \Delta b_{k,e} \bmod q$
- 13: Set $m_1 \leftarrow (\text{com}_e^{(i)}, (\Delta b_{k,e})_{k \in [M]})_{i \in [N], e \in [\tau]}$

Phase 2: Challenge a subset of the bundles to audit

- 1: Compute challenge hash: $h_1 \leftarrow H_1(\text{salt}, m_1, \text{attrs})$
- 2: Compute $C \leftarrow \text{Expand}(\text{salt}, h_1)$ where $C \subset \{1, \dots, M\}$, $|C| = \sigma$

Phase 3: Generate keypair, simulate MPC, commit to views, and open the audited bundles

- 1: Compute $(S, E) \leftarrow \text{Arrange}(\{v_1, \dots, v_M\}, C)$
- 2: Choose random seed_A , derive $A \in \mathbb{Z}_q^{n \times m}$ as in FrodoKEM
- 3: Compute public key: $B \leftarrow AS + E \bmod q$
- 4: **for** each parallel repetition e **do**
- 5: **for** each party i **do**
- 6: Compute $(S_e^{(i)}, E_e^{(i)}) \leftarrow \text{Arrange}((b_{k,e}^{(i)})_{k \in [M]}, C)$
- 7: Compute $B_e^{(i)} \leftarrow AS_e^{(i)} + E_e^{(i)} \bmod q$
- 8: Open audited bundles: set $X_e \leftarrow \{b_{k,e}^{(i)} : k \notin C, i \in [N]\}_{e \in [\tau]}$
- 9: Set $m_2 \leftarrow (B, (B_e^{(1)}, \dots, B_e^{(N)})_{e \in [\tau]}, (X_e)_{e \in [\tau]})$

Phase 4: Challenge the MPC simulation

- 1: Compute challenge hash: $h_2 \leftarrow H_2(\text{salt}, h_1, m_2)$
- 2: Compute $(i_1^*, \dots, i_\tau^*) \leftarrow \text{Expand}(\text{salt}, h_2)$ where $i_e^* \in [N]$

Step 5: Open the MPC views and output the key pair and proof

- 1: **for** each parallel repetition e **do**
- 2: $\text{seeds}_e \leftarrow$ nodes needed to compute $\{\text{seed}_e^{(i)}, i \neq i_e^*\}$
- 3: Output $\text{pk} = (B, \text{seed}_A)$, $\text{sk} = (S, E)$ and the proof π :
 $(h_1, h_2, \text{salt}, (\text{seeds}_e, \text{com}_e^{(i_e^*)}, (\Delta b_{k,e})_{k \in [M], e \in [\tau]}, (v_k)_{k \notin C})$

Figure 3: Combined proof and key generation scheme, prover operations. Commit is a commitment scheme. See paragraph “Helper functions” for description of functions Arrange and Sample, and pseudorandom generators Expand and ExpandTape.

is exponential. For the practicality of our protocol, we want M to be as small as possible. In this section we will show that it is sufficient for the prover to convince the verifier that *most* of the secret values are small, and that this is possible for practical choices of M .

Suppose Bob is honest and has generated $\text{pk}_B = (B, \text{seed}_A)$, but Alice is malicious and wants to create a PoP for pk_B . Therefore Alice aims to demonstrate knowledge of a pair (S', E') satisfying the relation $B = AS' + E'$. If there are no size restrictions on the entries of S' or E' , then it is easy to find S' and E' that are consistent with B . If Alice's malicious proof verifies, the verifier is convinced that most of the values in (S', E') are small. But Lemma 4 shows

```

cPoP.Vf(pk, π, attrs):
1: Parse pk and π, expand h1 and h2 as defined in Figure 3
2: Derive A ∈ ℤqn×m from seedA
3: for each parallel repetition e do
4:   for each party i ∈ [N], i ≠ ie* do
5:     Compute come(i) ← Commit(salt, e, i, seede(i))
6:     Expand tape: tapee(i) ← ExpandTape(salt, e, i, seede(i))
7:     Compute shares: (bk,1(i), ..., bk,τ(i)) ← Sample(tapee(i)).
8:     if i = 1, update P1's share: bk,e(1) ← bk,e(1) + Δbk,e mod q
9:     Compute (Se(i), Ee(i)) ← Arrange((bk,e(i))k∈[M], C)
10:    Compute Be(i) ← ASe(i) + Ee(i) mod q
11:    Compute missing shares Bj(i*) ← B - ∑i≠i* Bj(i) mod q
12: for k ∉ C do                                ▶ Audit opened bundles
13:   Abort if vk does not meet the range criteria given by χ
14:   for each parallel repetition e do
15:     Recompute missing party's share:
16:       
$$b_{k,e}^{(i_e^*)} \leftarrow v_k - \sum_{i \in [N] \setminus i_e^*} b_{k,j}^{(i)} \bmod q$$

17:     if ie* = 1, bk,e(ie*) ← bk,e(ie*) + Δbk,e mod q
18:     Reconstruct Xe from bk,e(i) as in Figure 3
19:   Compute m1 and m2 as defined in Figure 3
20:   Compute h'1 ← H1(salt, m1, attrs) and h'2 ← H2(salt, h'1, m2)
21:   Output accept if h'1 = h1 and h'2 = h2, else output reject

```

Figure 4: Combined proof and key generation scheme, verifier operations.

that, since the honest secret key Bob generated has small (S, E), then it is *unique* (unconditionally, with overwhelming probability). Therefore, if Alice's proof verifies, we must have (S', E') = (S, E) and this implies that (S', E') are in fact all small values.

The following lemma bounds the probability that, for a well-formed FrodoKEM key, there exists a second solution (S', E') where every entry is small, or at most γ entries total are not small.

LEMMA 4 (UNIQUENESS OF SMALL FRODOKEM SOLUTIONS). *Let $q = 2^D$, and let D, n, \bar{n} be positive integers. Let $A \leftarrow_s \mathbb{Z}_q^{n \times n}$ be selected uniformly at random, and let $S, E \leftarrow_s \chi_q^{n \times \bar{n}}$ have each entry be generated independently according to the FrodoKEM error distribution χ_q . Let β be the maximum value in the support of χ_q . Let $B = AS + E$. Then the probability that there are values $S', E' \in \mathbb{Z}_q^{n \times \bar{n}}$ such that each entry of (S', E') has absolute value at most β , that $B = AS' + E'$, and that (S', E') ≠ (S, E), is at most*

$$\bar{n}(1 - 2^{-n})(2\beta + 1)^n \cdot \sum_{t=0}^{D-1} \left(\left\lceil \frac{2\beta + 1}{2^t} \right\rceil / q \right)^n. \quad (2)$$

Furthermore, if the size requirement is relaxed for at most γ entries of S' and E', then this probability is at most

$$\bar{n}(1 - 2^{-n})(2\beta + 1)^{n-\gamma} q^{\gamma-n} \cdot \sum_{t=0}^{D-1} \left\lceil \frac{2\beta + 1}{2^t} \right\rceil^n. \quad (3)$$

PROOF. Consider the first column of S, the first row of A, the first entry of the first row of B, and the first entry of the first column of E, denoted s, a, b, and e respectively. Let $s' \in \mathbb{Z}_q^n$ with small coefficients

(absolute value less than β). Let $e' = b - \langle a, s' \rangle$. Then s' is a valid first column for a solution matrix S of the FrodoKEM relation if $|e'|$ is less than or equal to β . We can write $e' = \langle a, \bar{s} \rangle + e$, where $\bar{s} = s - s'$. For every $\bar{s} \neq 0$ (and hence $s' \neq s$), we will calculate the probability that $|e'|$ is less than or equal to β , and then bound the probability that there exists a second solution s' for the first column of the FrodoKEM relation using a union bound.

Note that if there is at least one odd entry s_j in \bar{s} , then $\langle a, \bar{s} \rangle$ is uniformly distributed over \mathbb{Z}_q , since $\gcd(s_j, q) = 1$, and so $a_j \cdot s_j$ is uniformly distributed. Further, this means that $\langle a, \bar{s} \rangle + e$ is also uniformly distributed. Thus $\Pr[|e'| \leq \beta] = \frac{2\beta+1}{q}$.

If there are no odd entries in \bar{s} , then this probability can be higher. This occurs for a fraction of possible values of size 2^{-n} . Then, if there is at least one entry that is not a multiple of 4, $\langle a, \bar{s} \rangle$ is distributed uniformly over even values in \mathbb{Z}_q . This gives us that $\Pr[|e'| \leq \beta] \leq 2 \cdot \left\lceil \frac{2\beta+1}{2} \right\rceil / q$. Continuing in this way for $t \in \{0, \dots, D-1\}$, we get that for a proportion of possible values for \bar{s} of size $2^{-tn}(1 - 2^{-n})$, we have $\Pr[|e'| \leq \beta] \leq 2^t \cdot \left\lceil \frac{2\beta+1}{2^t} \right\rceil / q$.

The probability that the above relation holds for all n rows of the matrix A (using the corresponding other entries of the first column of E) is then at most $\left(2^t \cdot \left\lceil \frac{2\beta+1}{2^t} \right\rceil / q \right)^n$. Performing a union bound over all $(2\beta + 1)^n$ possible values of \bar{s} gives the following upper bound on the probability that there exists a second solution s' for the first column:

$$\begin{aligned} & \sum_{t=0}^{D-1} 2^{-tn}(1 - 2^{-n})(2\beta + 1)^n \left(2^t \cdot \left\lceil \frac{2\beta + 1}{2^t} \right\rceil / q \right)^n \\ & = (1 - 2^{-n})(2\beta + 1)^n \cdot \sum_{t=0}^{D-1} \left(\left\lceil \frac{2\beta + 1}{2^t} \right\rceil / q \right)^n. \end{aligned}$$

Repeating this argument for all \bar{n} columns of S (and corresponding columns of E) gives an upper bound of \bar{n} times the probability above, which is Equation (2).

Now we consider what happens when we relax the size requirement for some values of s' and e' . Specifically, if γ_1 entries of the first column of E do not need to meet the size constraint and γ_2 entries of s' do not need to meet the size constraint, then the above union bound instead becomes

$$\begin{aligned} & (1 - 2^{-n})(2\beta + 1)^{n-\gamma_2} q^{\gamma_2} \cdot \sum_{t=0}^{D-1} 2^{-tn} \left(2^t \left\lceil \frac{2\beta + 1}{2^t} \right\rceil / q \right)^{n-\gamma_1} \\ & = (1 - 2^{-n})(2\beta + 1)^{n-\gamma_2} q^{\gamma_1+\gamma_2-n} \cdot \sum_{t=0}^{D-1} 2^{-t\gamma_1} \left\lceil \frac{2\beta + 1}{2^t} \right\rceil^{n-\gamma_1}. \end{aligned}$$

Let $\gamma = \gamma_1 + \gamma_2$. The choices of γ_1 and γ_2 maximizing this probability with a given γ will have $\gamma_1 = 0$ and $\gamma_2 = \gamma$. Again considering this result over all \bar{n} columns (taking a course upper bound by multiplying by \bar{n} , even though not all of the columns can have γ relaxed values simultaneously) of S gives us Equation (3). \square

4.3.1 Parameter Selection. We explain how we use Lemma 4 with an example, consider parameters for Frodo640: $n = 640$, $\bar{n} = 8$, $\beta = 12$, and $q = 2^{15}$. We choose γ to be the largest integer so that, as per the relaxed bound in Lemma 4, the probability of a non-unique solution is less than 2^{-128} . The result is $\gamma = 340$. Hence we can

relax up to 340 size constraints in an alternate solution, and the probability of the FrodoKEM solution S being non-unique is still less than 2^{-128} . We note that this probability holds unconditionally.

Consequently, we choose M in the FrodoKEM ZK-PoP protocol so that the probability that 341 or more invalid bundles all fall in the σ unaudited values is less than 2^{-128} . If γ is the number of invalid bundles, then the probability of all γ invalid bundles being in the σ unaudited bundles is $\binom{M-\gamma}{M-\sigma} / \binom{M}{M-\sigma}$. For Frodo640, where $\sigma = 10240$, the smallest value of M for which this bound is smaller than 2^{-128} , with $\gamma = 341$, is $M = 10240 + 2993$. We give other values of M and γ for FrodoKEM and Kyber in the full version [40].

The second group of parameters are chosen to prevent cheating in the MPCitH executions. Since the τ executions (each with N parties) are independent, the probability of successful cheating is $\frac{1}{N^\tau}$. For κ -bit security, we choose N and τ so this is at most $2^{-\kappa}$.

4.4 Security Analysis

We prove two lemmas about the protocol of Section 4.2, that roughly correspond to the two properties required for a secure cPoP scheme. Due to space constraints, proofs of lemmas from this section appear in the full version [40].

The first is honest verifier zero-knowledge, which corresponds to the ZK property of cPoPs (Definition 3). For this lemma we require that ExpandTape is a secure pseudorandom generator (PRG) as defined in, e.g., [14, Definition 3.1], implemented with a random oracle. As our implementation uses SHAKE [58] to implement ExpandTape this is a standard assumption. For the seed tree construction used to derive the per-party seeds, we (informally) must assume that after revealing $N - 1$ of N seeds, the remaining seed is hidden to a computationally bounded adversary. This is shown to hold when the tree is constructed with a random oracle in [20, Section 6.3], and again our implementation uses SHAKE.

LEMMA 5. *The non-interactive cPoP protocol of Section 4.2 is zero-knowledge in the random oracle model.*

Straight-line extractability. Next we prove that proofs created by Section 4.2 are *straight-line extractable* in the ROM, meaning that we can extract the secret key from a successful prover. In the context of a cPoP, the adversary may create many proofs that verify, however, if they create one for an existing public key, our proof of cPoP security (Theorem 7) will use this property to recover the secret key, reducing cPoP-UF to key recovery (which is a search LWE problem for FrodoKEM).

Our proof is loosely based on the UF-KOA security proof for the Rainier signature scheme [29], and is in parts identical. (As Rainier is based on the Banquet scheme [6], its security analysis is in turn similar to Banquet's.) Rainier is also based on a 5-round MPCitH proof, and the KOA security analysis implicitly defines an extractor in the ROM: when the adversary creates a forgery (a valid proof for a challenge public key), the proof extracts the secret key from the query history, in order to reduce KOA security to key recovery. This proof strategy is common for the analysis of MPCitH-based proofs, first described in [46, §3.1]. The basic idea is that because we've chosen our parameters correctly, if π verifies, then at least one of the τ parallel repetitions was executed honestly. Let index e be one such repetition. From the query history, the extractor algorithm Ext

can find the preimage of $\text{com}_e^{(i^*)}$, which contains $\text{seed}_e^{(i^*)}$. Then Ext has the shares of all N parties and can recover (S, E) .

We formally define straight-line extractability (SLE) for cPoPs in the full version. SLE says that for any proof and key pair (pk, π, attrs) output by \mathcal{A} such that $\text{cPoP.Vf}(pk, \pi, \text{attrs}) = 1$, there exists an extractor $\text{Ext}_{\text{Commit}, H_1, H_2}(pk, \pi, \text{attrs})$ that recovers sk such that $R(pk, sk) = 1$, where R relates public and secret keys. Ext simulates the random oracles used by \mathcal{A} and fails with bounded probability.

LEMMA 6. *Let Commit, H_1 , and H_2 be modeled as random oracles, Expand be modeled as a random function, and let cPoP be the cPoP scheme given in Figures 3 and 4 with parameters (N, τ) . Let $\mathcal{A}^{\text{Commit}, H_1, H_2}$ be an adversary that makes a total of Q random oracle queries. Then cPoP is straight-line extractable with*

$$\epsilon_{\text{sle}} \leq \frac{(\tau N + 1)Q^2}{2^{2\kappa}} + 1/2^\kappa$$

when parameters are chosen as described in Section 4.3.1.

Note that when sk is not unique for a given pk the extractor of Lemma 6 might not output the same sk used by \mathcal{A} . However, when pk is honestly generated, the analysis of Section 4.3 ensures that sk is unique and Lemma 6 ensures that Ext will extract it.

We now come to the main theorem of this section, that uses the previous two lemmas to prove that our new PoP is secure. We also require that the KEM be KEM-SIM secure; we define this property in the full version [40] and prove that it holds for KEMs constructed with the FO transform (including FrodoKEM and Kyber). We say that the key generation function of a KEM is a one-way function (as defined in [39, Section 2.2]) if it hard to recover sk from pk , such that $R(sk, pk) = 1$. Note that IND-CPA security of the KEM implies one-wayness of the key generation function.

THEOREM 7. *If KEM.KeyGen is a one-way function, and KEM is KEM-SIM secure, then the cPoP construction of Section 4.2 is a secure cPoP in the random oracle model, when KEM.Decaps is the auxiliary secret key usage algorithm.*

PROOF. The cPoP security definition Definition 2 has three parts: zero-knowledge, correctness, and unforgeability.

The ZK property (Definition 3) follows immediately from our ZK lemma, when using the simulator in Lemma 5 in $\text{Exp}_{\text{cPoP}}^{\text{ZK}}$ (Figure 2), \mathcal{A} 's distinguishing advantage is negligible, bounded by ϵ_{zk} . Note that in combined key generation and PoP schemes, \mathcal{A} gets only one proof per key pair.

The correctness property is immediate if, in the cPoP proof, P samples (v_1, \dots, v_M) using the same procedure as in KEM.KeyGen, which is the case for our construction in Figure 3.

Now for the UF property (Definition 2). Let \mathcal{A} be a cPoP attacker in the security experiment $\text{Exp}_{\text{cPoP}, \text{Aux}}^{\text{UF}}$ given in Figure 1. We construct \mathcal{B} , an algorithm that uses \mathcal{A} as a subroutine in the ROM to invert KEM.KeyGen. Therefore \mathcal{B} must implement the random oracles H_1, H_2 , and Commit for \mathcal{A} . Algorithm \mathcal{B} is initialized with a public key pk output by KEM.KeyGen, and must output sk such that $R(pk, sk) = 1$. Algorithm \mathcal{A} is initialized with nothing, and starts by outputting attrs .

We describe a sequence of games, where \mathcal{B} starts by implementing $\text{Exp}_{\text{cPoP}, \text{Aux}}^{\text{UF}}(\mathcal{A})$ as given in Figure 1 (this is Game 0), then in the last game \mathcal{B} inverts KEM.KeyGen if \mathcal{A} succeeds.

In Game 1, \mathcal{B} replaces calls to Aux , which are KEM.Decaps queries, with simulated Decaps queries, using the simulator given by KEM-SIM security of KEM . Thus $G_1 - G_0 \leq \epsilon_{\text{KEM-SIM}}$.

In Game 2, \mathcal{B} no longer computes $(pk, sk, \pi) \leftarrow \text{cPoP.KPG}(\text{attrs})$; instead \mathcal{B} computes $(pk, sk) \leftarrow \text{KEM.KeyGen}()$, then uses the $\text{ZK Sim}(pk, \text{attrs})$ from Lemma 5 to simulate π . As pk is identically distributed in cPoP and KEM.KeyGen , this part of the change is identical in both games. The proof π is indistinguishable except with probability ϵ_{ZK} .

In Game 3, \mathcal{B} no longer calls KEM.KeyGen in order to generate (pk, sk) . As of Game 2, only pk is needed, and \mathcal{B} gets a challenge pk in the OWF security game for KEM.KeyGen .

Now suppose \mathcal{A} wins in Game 3, i.e., outputs (π', attrs') such that $\text{cPoP.Vf}(pk, \pi', \text{attrs}') = 1$. \mathcal{B} uses the straight-line extractor Ext of Lemma 6 to extract sk' from \mathcal{A} such that $R(pk, sk') = 1$, and outputs sk' in the OWF security game for KEM.KeyGen . Therefore, when \mathcal{A} wins $\text{Exp}_{\text{cPoP, Aux}}^{\text{UF}}$, \mathcal{B} inverts KEM.KeyGen .

Conclusion. We have shown that the success probability for all adversaries \mathcal{A} against the cPoP scheme is bounded by $2\epsilon_{\text{ZK}} + q\epsilon_{\text{KEM-SIM}} + \epsilon_{\text{OW-KeyGen}} + \epsilon_{\text{Sle}}$, where q is the number of Aux queries made by \mathcal{A} . The quantity ϵ_{ZK} was shown to be negligible in Lemma 5, $\epsilon_{\text{KEM-SIM}}$ is given in the full version [40] for a class of KEMs constructed with the FO transform (including FrodoKEM and Kyber), and ϵ_{Sle} is given for our construction in Lemma 6. \square

Theorem 7 shows that the KEM does not undermine the security of the cPoP . In the full version [40] we consider the other direction and show that adding the cPoP does not undermine security of the KEM . We show IND-CCA and KEM-SIM security are preserved when KeyGen is replaced with the combined key and proof generation function and the proof is output with the public key. Security largely follows from the zero-knowledge property of the cPoP .

4.4.1 Resistance to quantum attacks. In the above analysis, we have assumed a classical attacker. A proof in the quantum random oracle model (QROM) would provide additional assurance against quantum attacks. Recent work [32] gives a way to prove straight line (or online) extractability of commit-and-open NIZK proofs in the QROM . Since our PoP is a commit-and-open protocol, and our extraction algorithm is simply reading the RO query histories, the techniques seem directly applicable to prove a QROM analog of Lemma 6. Our ZK result (Lemma 5), relies crucially on programming the random oracle so that Sim can know the challenge in advance. This was considered in [30, 31] in the context of signatures schemes constructed from 3- and 5-round ID schemes using the Fiat-Shamir transform. Since our combined generation and verification construction is not based on an ID scheme, these generic results cannot be immediately applied, however they appear to be the closest in the the QROM literature.

5 APPLICATION TO KYBER

Now we consider proof of possession via verifiable generation for Kyber instead of FrodoKEM . Kyber 's structure is similar to FrodoKEM , but Kyber is based on module LWE instead of plain LWE , so we need a new version of Lemma 4 to choose parameters.

LEMMA 8 (UNIQUENESS OF SMALL KYBER SOLUTIONS). *Let $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where $X^n + 1$ is the 2^n -th cyclotomic polynomial*

and q is prime. Let $\mathbf{A} \in R_q^{\ell \times \ell}$ be selected uniformly at random, and let $\mathbf{s} \in R_q^\ell$ and $\mathbf{e} \in R_q^\ell$ be sampled from a centered binomial distribution as in the Kyber specification, so that the absolute values of the coefficients of the polynomials in \mathbf{s} and \mathbf{e} are at most β . Let $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$. Then the probability that there are values $\mathbf{s}', \mathbf{e}' \in R_q^\ell$, with coefficients whose absolute values are at most β such that $\mathbf{b} = \mathbf{A}\mathbf{s}' + \mathbf{e}'$ and $(\mathbf{s}', \mathbf{e}') \neq (\mathbf{s}, \mathbf{e})$ is at most $(2\beta + 1)^{\ell n} \left(\frac{2\beta+1}{q}\right)^{\ell n}$. Furthermore, if the size requirement is relaxed for some γ coefficients of \mathbf{s}' and \mathbf{e}' , then the probability is $(2\beta + 1)^{\ell n} \left(\frac{2\beta+1}{q}\right)^{\ell n - \gamma}$.

See the full version [40] for the proof.

To generate a Kyber key pair along with a proof of possession, we can use a slightly modified version of our cPoP from Section 4. The differences will be that the secret values are sampled differently, σ is different, and vectors are replaced by polynomials where appropriate. We then choose parameters as in Section 4.3.1, where the main difference is the choice of M , calculated using Lemma 8 instead of Lemma 4. Parameters (N, τ) are the same for Kyber , and the values of (M, σ) are given in the full version [40].

6 IMPLEMENTATION AND EVALUATION

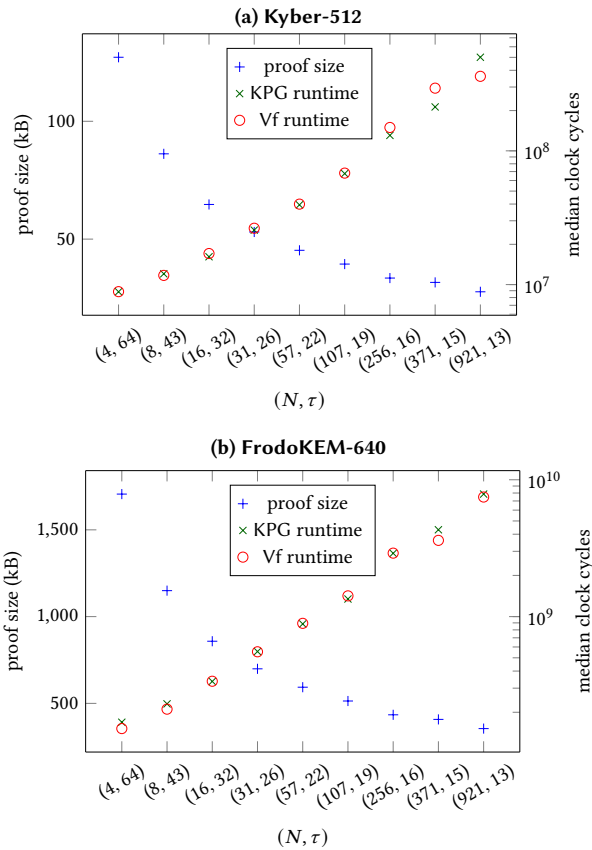
We now discuss performance of the key and proof generation and proof verification, and resulting proof sizes, depending on the number of parties N and the number of parallel executions τ .

Parameter sizes Recall that we must choose (N, τ) such that $N^\tau \geq 2^\kappa$ for κ -bit security, and note that there are many valid pairs for each κ . As with all MPCitH proofs with flexible choice of N , we have a speed-size tradeoff: larger N gives shorter proofs that are slower to create and verify (as there are more parties to simulate), while smaller N has larger and faster proofs. The curve is very steep; a small change in size results in a large change in speed (we plot these curves for our implementation in Figure 5).

We implemented our construction for all FrodoKEM and Kyber parameter sets by extending their existing public C code. For Kyber , we use the already highly optimized AVX2 code base, whereas the FrodoKEM code base has no hand-crafted AVX2 optimized code, but rather exploits the compiler's ability to auto-vectorize code. The benchmarks have been conducted on an Intel Core i7-8565U CPU running at up to 4.6 GHz, compiled with gcc 11.2.0, and with the CPU scaling governor set to performance. Figure 5 shows the median cycle counts for KPG and Vf as well as the median proof size, all over 100 executions. To obtain reliable median cycle counts for Vf , we measure the median cycle count of 100 executions of KPG and Vf . Each of these median values is then subtracted by the respective median cycle count of KPG .

Profile of execution For Kyber512 approximately 51% of the time is spent hashing (out of which 74% is parallelized) and 26% on sampling uniform randomness mod q , which is partially parallelized. Arithmetic operations take 7% of the overall time. Furthermore, 16% is spent on Arrange , which is not parallelized.

Optimizations For our implementations, we use a number of optimizations. In Kyber , we use dedicated AVX2 intrinsics to vectorize the computation of $\Delta b_{k,e}$ and the computation of the missing shares $\{b_{e,k}^{(i_e)}\}_{k \in C}$. For the polynomial arithmetic, we use the already existing optimized functions. Apart from that, the main goal is to

Figure 5: KPG and Vf performance and proof size for 128-bit security

speed up the costly hashing operations. We instantiate all hash and expansion functions with SHAKE-128 for $\kappa = 128$ and SHAKE-256 for $\kappa \in \{192, 256\}$ and apply domain separation where necessary. We use the existing AVX2-enabled 4-times parallel hashing where possible: to compute the commitments $\text{com}_e^{(i)}$ in KPG and $\text{com}_e^{(i)}$ in Vf, respectively; and to expand the seeds both in KPG and Vf. Additionally, we hash the values $B_e^{(i)}$ in parallel, and then add their hashes to m_2 rather than the shares themselves. Similarly, we hash the audited bundles X_e for each party in parallel, and then add their hashes to m_2 rather than the audited bundles themselves.

More details on performance and proof size in higher security cases can be found in the full version [40].

7 CONCLUSION AND DISCUSSION

From our construction, we can achieve a proof of possession for KEM keys that can be used in a non-interactive certificate enrollment process, similar to Certificate Signing Requests.

It is the case that our combined proof and key generation method means only a single proof can be generated. To authenticate other events in the certificate lifecycle, we cannot use a proof from the same verifiable generation approach (since proof generation happens concurrently with key generation), but fortunately certificate

lifecycle protocols like ACME already allow for revocation events to be signed by a separate account key [5, §7.6]. Furthermore, by using the attributes field we can bind a signature public key to the PoP and use it for subsequent PoPs (or revocation), as a work-around.

It is not required to know the authority or to obtain a nonce from them in advance, since this would break the desirable non-interactive property of CSRs. For example, the ACME Protocol [5] for certificate enrolment does not require that CSR contents are bound to the protocol or CA, and ACME doesn't include any provisions preventing CSR replays (which is often considered a feature, e.g. where the ACME client is run outside of the production network and does not have access to the private key).

Our construction allows arbitrary attributes to be bound to the key generation, which can be used to incorporate the standard fields from a CSR or more, binding the key generation to its intended use.

Compared to existing techniques to prove knowledge of LWE secrets, our MPCitH-based approach for proof of possession via verifiable generation can be configured to both run in reasonable time *and* achieve reasonably low proof sizes; see Table 1 for some comparisons. Additionally, the flexibility in parameter choice allows adjustments to meet use case requirements. For CSRs, which are usually generated infrequently, a runtime on the order of a second might be acceptable, which helps to keep communication cost low while still providing reasonable user experience. On the other side, when a larger proof is acceptable, it is possible to run KPG and Vf in less than 10 ms for Kyber512.

Furthermore, our construction scales well for higher parameter sets with $\kappa \in \{192, 256\}$. To the best of our knowledge, we are the first to give a practical proof of possession with parameters and an implementation for lattice secrets for security levels above 128 bits.

We have demonstrated our technique for FrodoKEM and Kyber. For the remaining NIST Round 3 lattice-based KEMs, SABER [25] and NTRU [22], our approach may be applied in some form, since MPCitH is so flexible. However, doing so efficiently may require some additional research, due to differences in the constructions. SABER is close in structure to FrodoKEM and Kyber, but its public key computation includes a right-shift that removes low-order bits from the public key, which is not conducive to efficient implementation in our MPCitH approach. NTRU has more differences in its structure, and key generation includes an inversion operation in the ring which can be implemented with an MPCitH proof, albeit with a different MPC protocol (e.g., see [6] for approaches to implementing the AES S-box, which is a field inversion). To follow our approach, one would also need to prove analogs of Lemmas 4 and 8 about uniqueness of small solutions.

ACKNOWLEDGMENTS

D.S. was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery grants RGPIN-2016-05146 and RGPIN-2022-03187 and an NLnet Assure grant for the project "Standardizing KEMTLS". T.G. and G.L. were supported by the German Research Foundation under Germany's Excellence Strategy – EXC 2092 CASA – 390781972, through the H2020 project PROMETHEUS (grant agreement ID 780701), and by the Federal Ministry of Education and Research of Germany through the QuantumRISC (16KIS1038) and PQC4Med (16KIS1044) projects.

REFERENCES

- [1] Carlisle Adams, Stephen Farrell, Tomi Kause, and Tero Mononen. 2005. RFC 4210: Internet X.509 public key infrastructure certificate management protocol (CMP).
- [2] Carlisle Adams and Steve Lloyd. 1999. *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Sams Publishing.
- [3] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkatasubramanian. 2017. Liger: Lightweight Sublinear Arguments Without a Trusted Setup. In *ACM CCS 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, 2087–2104. <https://doi.org/10.1145/3133956.3134104>
- [4] Frederico Araujo, Kevin W. Hamlen, Sebastian Biedermann, and Stefan Katzenbeisser. 2014. From Patches to Honey-Patches: Lightweight Attacker Misdirection, Deception, and Disinformation. In *ACM CCS 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM Press, 942–953. <https://doi.org/10.1145/2660267.2660329>
- [5] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. 2019. RFC 8555: Automatic Certificate Management Environment (ACME).
- [6] Carsten Baum, Cyprien de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. 2021. Banquet: Short and Fast Signatures from AES. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, 266–297. https://doi.org/10.1007/978-3-030-75245-3_11
- [7] Carsten Baum and Ariel Nof. 2020. Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography. In *PKC 2020, Part I (LNCS, Vol. 12110)*, Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas (Eds.). Springer, Heidelberg, 495–526. https://doi.org/10.1007/978-3-030-45374-9_17
- [8] Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS '93*, Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby (Eds.). ACM Press, 62–73. <https://doi.org/10.1145/168588.168596>
- [9] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. 2019. Aurora: Transparent Succinct Arguments for R1CS. In *EUROCRYPT 2019, Part I (LNCS, Vol. 11476)*, Yuval Ishai and Vincent Rijmen (Eds.). Springer, Heidelberg, 103–128. https://doi.org/10.1007/978-3-030-17653-2_4
- [10] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2012. High-speed high-security signatures. *Journal of Cryptographic Engineering* 2, 2 (Sept. 2012), 77–89. <https://doi.org/10.1007/s13389-012-0027-1>
- [11] Ward Beullens. 2020. Sigma Protocols for MQ, PKP and SIS, and Fishy Signature Schemes. In *EUROCRYPT 2020, Part III (LNCS, Vol. 12107)*, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, 183–211. https://doi.org/10.1007/978-3-030-45727-3_7
- [12] Ward Beullens and Cyprien de Saint Guilhem. 2020. LegRoast: Efficient Post-quantum Signatures from the Legendre PRF. In *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, Jintai Ding and Jean-Pierre Tillich (Eds.). Springer, Heidelberg, 130–150. https://doi.org/10.1007/978-3-030-44223-1_8
- [13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS 2012*, Shafi Goldwasser (Ed.). ACM, 326–349. <https://doi.org/10.1145/2090236.2090263>
- [14] Dan Boneh and Victor Shoup. 2020. A Graduate Course in Applied Cryptography. Available at <https://crypto.stanford.edu/~dabo/cryptobook/>.
- [15] Jonathan Bootle, Vadim Lyubashevsky, and Gregor Seiler. 2019. Algebraic Techniques for Short(er) Exact Lattice-Based Zero-Knowledge Proofs. In *CRYPTO 2019, Part I (LNCS, Vol. 11692)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 176–202. https://doi.org/10.1007/978-3-030-26948-7_7
- [16] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 353–367. <https://doi.org/10.1109/EuroSP.2018.00032>
- [17] Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. 2016. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. In *ACM CCS 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM Press, 1006–1018. <https://doi.org/10.1145/2976749.2978425>
- [18] Cecilia Boschini, Jan Camenisch, Max Ovsiankin, and Nicholas Spooner. 2020. Efficient Post-quantum SNARKs for RSIS and RLWE and Their Applications to Privacy. In *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, Jintai Ding and Jean-Pierre Tillich (Eds.). Springer, Heidelberg, 247–267. https://doi.org/10.1007/978-3-030-44223-1_14
- [19] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short Proofs for Confidential Transactions and More. In *2018 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 315–334. <https://doi.org/10.1109/SP.2018.00020>
- [20] Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, and Greg Zaverucha. 2020. The Picnic Signature Scheme Design Document (version 2.2). <https://github.com/microsoft/Picnic/tree/master/spec>
- [21] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. 2017. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In *ACM CCS 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM Press, 1825–1842. <https://doi.org/10.1145/3133956.3133997>
- [22] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hulsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. 2020. *NTRU*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions>.
- [23] Vincent Cheval, Véronique Cortier, and Bogdan Warinschi. 2017. Secure Composition of PKIs with Public Key Protocols. In *CSF 2017 Computer Security Foundations Symposium*, Boris Köpf and Steve Chong (Eds.). IEEE Computer Society Press, 144–158. <https://doi.org/10.1109/CSF.2017.28>
- [24] Santosh Chokhani, W Ford, R Sabett, C Merrill, and S Wu. 2003. RFC 3647: Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework.
- [25] Jan-Pieter D’Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. 2020. *SABER*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions>.
- [26] Cyprien de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. 2019. BBQ: Using AES in Picnic Signatures. In *SAC 2019 (LNCS, Vol. 11959)*, Kenneth G. Paterson and Douglas Stebila (Eds.). Springer, Heidelberg, 669–692. https://doi.org/10.1007/978-3-030-38471-5_27
- [27] Cyprien de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. 2021. Limbo: Efficient Zero-knowledge MPCitH-based Arguments. In *ACM CCS 2021*, Giovanni Vigna and Elaine Shi (Eds.). ACM Press, 3022–3036. <https://doi.org/10.1145/3460120.3484595>
- [28] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. 2019. Short Discrete Log Proofs for FHE and Ring-LWE Ciphertexts. In *PKC 2019, Part I (LNCS, Vol. 11442)*, Dongdai Lin and Kazuo Sako (Eds.). Springer, Heidelberg, 344–373. https://doi.org/10.1007/978-3-030-17253-4_12
- [29] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schafneger, and Greg Zaverucha. 2021. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In *ACM CCS 2022*, Cas Cremers and Elaine Shi (Eds.). ACM Press. Available at <https://eprint.iacr.org/2021/692>.
- [30] Jelle Don, Serge Fehr, and Christian Majenz. 2020. The Measure-and-Reprogram Technique 2.0: Multi-round Fiat-Shamir and More. In *CRYPTO 2020, Part III (LNCS, Vol. 12172)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 602–631. https://doi.org/10.1007/978-3-030-56877-1_21
- [31] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2019. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In *CRYPTO 2019, Part II (LNCS, Vol. 11693)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 356–383. https://doi.org/10.1007/978-3-030-26951-7_13
- [32] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. 2022. Efficient NIZKs and Signatures from Commit-and-Open Protocols in the QROM. Cryptology ePrint Archive, Report 2022/270. <https://eprint.iacr.org/2022/270>.
- [33] Daniel Escudero, Satrajit Ghosh, Marcel Keller, Rahul Rachuri, and Peter Scholl. 2020. Improved Primitives for MPC over Mixed Arithmetic-Binary Circuits. In *CRYPTO 2020, Part II (LNCS, Vol. 12171)*, Daniele Micciancio and Thomas Ristenpart (Eds.). Springer, Heidelberg, 823–852. https://doi.org/10.1007/978-3-030-56880-1_29
- [34] Muhammed F. Esgin, Ngoc Khanh Nguyen, and Gregor Seiler. 2020. Practical Exact Proofs from Lattices: New Techniques to Exploit Fully-Splitting Rings. In *ASIACRYPT 2020, Part II (LNCS, Vol. 12492)*, Shiho Moriai and Huaxiong Wang (Eds.). Springer, Heidelberg, 259–288. https://doi.org/10.1007/978-3-030-64834-3_9
- [35] Thibault Feneuil, Antoine Joux, and Matthieu Rivain. 2022. Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs. Cryptology ePrint Archive, Report 2022/188. <https://eprint.iacr.org/2022/188>.
- [36] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO '86 (LNCS, Vol. 263)*, Andrew M. Odlyzko (Ed.). Springer, Heidelberg, 186–194. https://doi.org/10.1007/3-540-47721-7_12
- [37] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *CRYPTO '99 (LNCS, Vol. 1666)*, Michael J. Wiener (Ed.). Springer, Heidelberg, 537–554. https://doi.org/10.1007/3-540-48405-1_34
- [38] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. 2016. ZKBoo: Faster Zero-Knowledge for Boolean Circuits. In *USENIX Security 2016*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 1069–1083.

- [39] Oded Goldreich. 2007. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press.
- [40] Tim Güneysu, Philip Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. 2022. Proof-of-possession for KEM certificates using verifiable generation. Cryptology ePrint Archive, Report 2022/703. <https://eprint.iacr.org/2022/703>.
- [41] Peter Gutmann. 2020. RFC 8894: Simple Certificate Enrolment Protocol.
- [42] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. 2017. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *TCC 2017, Part I (LNCS, Vol. 10677)*, Yael Kalai and Leonid Reyzin (Eds.). Springer, Heidelberg, 341–371. https://doi.org/10.1007/978-3-319-70500-2_12
- [43] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2009. Zero-Knowledge Proofs from Secure Multiparty Computation. *SIAM J. Comput.* 39, 3 (2009), 1121–1152. <https://doi.org/10.1137/080725398>
- [44] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. 2018. IND-CCA-Secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. In *CRYPTO 2018, Part III (LNCS, Vol. 10993)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 96–125. https://doi.org/10.1007/978-3-319-96878-0_4
- [45] Daniel Kales and Greg Zaverucha. 2020. Improving the Performance of the Picnic Signature Scheme. *IACR TCHES 2020*, 4 (2020), 154–188. <https://doi.org/10.13154/tches.v2020.i4.154-188> <https://tches.iacr.org/index.php/TCHES/article/view/8680>.
- [46] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. 2018. Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In *ACM CCS 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM Press, 525–537. <https://doi.org/10.1145/3243734.3243805>
- [47] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. 2008. Concurrently Secure Identification Schemes Based on the Worst-Case Hardness of Lattice Problems. In *ASIACRYPT 2008 (LNCS, Vol. 5350)*, Josef Pieprzyk (Ed.). Springer, Heidelberg, 372–389. https://doi.org/10.1007/978-3-540-89255-7_23
- [48] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* 75 (2015), 565–599. Issue 3. <https://doi.org/10.1007/s10623-014-9938-4>
- [49] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. 2018. Lattice-Based Zero-Knowledge Arguments for Integer Relations. In *CRYPTO 2018, Part II (LNCS, Vol. 10992)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 700–732. https://doi.org/10.1007/978-3-319-96881-0_24
- [50] Richard Lindner and Chris Peikert. 2011. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *CT-RSA 2011 (LNCS, Vol. 6558)*, Aggelos Kiyias (Ed.). Springer, Heidelberg, 319–339. https://doi.org/10.1007/978-3-642-19074-2_21
- [51] San Ling, Khoa Nguyen, Damien Stehlé, and Huaxiong Wang. 2013. Improved Zero-Knowledge Proofs of Knowledge for the ISIS Problem, and Applications. In *PKC 2013 (LNCS, Vol. 7778)*, Kaoru Kurosawa and Goichiro Hanaoka (Eds.). Springer, Heidelberg, 107–124. https://doi.org/10.1007/978-3-642-36362-7_8
- [52] Vadim Lyubashevsky. 2008. Lattice-Based Identification Schemes Secure Under Active Attacks. In *PKC 2008 (LNCS, Vol. 4939)*, Ronald Cramer (Ed.). Springer, Heidelberg, 162–179. https://doi.org/10.1007/978-3-540-78440-1_10
- [53] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. 2022. Efficient Lattice-Based Blind Signatures via Gaussian One-Time Signatures. In *PKC 2022, Part II (LNCS, Vol. 13178)*, Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe (Eds.). Springer, Heidelberg, 498–527. https://doi.org/10.1007/978-3-030-97131-1_17
- [54] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2020. Practical Lattice-Based Zero-Knowledge Proofs for Integer Relations. In *ACM CCS 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 1051–1070. <https://doi.org/10.1145/3372297.3417894>
- [55] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. 2021. Shorter Lattice-Based Zero-Knowledge Proofs via One-Time Commitments. In *PKC 2021, Part I (LNCS, Vol. 12710)*, Juan Garay (Ed.). Springer, Heidelberg, 215–241. https://doi.org/10.1007/978-3-030-75245-3_9
- [56] Daniele Micciancio and Salil P. Vadhan. 2003. Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More. In *CRYPTO 2003 (LNCS, Vol. 2729)*, Dan Boneh (Ed.). Springer, Heidelberg, 282–298. https://doi.org/10.1007/978-3-540-45146-4_17
- [57] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. 2020. *FrodoKEM*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions>.
- [58] NIST. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standards and Technology (NIST), FIPS PUB 202, U.S. Department of Commerce.
- [59] Magnus Nyström and Burt Kaliski. 2000. RFC 2986: PKCS #10: Certification Request Syntax Specification Version 1.7.
- [60] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. 2019. *NewHope*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions>.
- [61] Max Pritikin, Peter Yee, and Dan Harkins. 2013. RFC 7030: Enrollment over secure transport.
- [62] Oded Regev. 2003. New lattice based cryptographic constructions. In *35th ACM STOC*. ACM Press, 407–416. <https://doi.org/10.1145/780542.780603>
- [63] Eric Rescorla. 2018. RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3.
- [64] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. 2020. *CRYSTALS-KYBER*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions>.
- [65] Peter Schwabe, Douglas Stebila, and Thom Wiggers. 2020. Post-Quantum TLS Without Handshake Signatures. In *ACM CCS 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM Press, 1461–1480. <https://doi.org/10.1145/3372297.3423350>
- [66] Jacques Stern. 2006. A New Paradigm for Public Key Identification. *IEEE Trans. Inf. Theor.* 42, 6 (sep 2006), 1757–1768. <https://doi.org/10.1109/18.556672>
- [67] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladimir Kolesnikov, and Daniel Kales. 2020. *Picnic*. Technical Report. National Institute of Standards and Technology. Available at <https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions>.