Quantum-Safe Public Key Blinding from MPC-in-the-Head Signature Schemes

Sathvika Balumuri¹, Edward Eaton¹ and Philippe Lamontagne^{1,2}

¹ National Research Council Canada
² Université de Montréal, Canada

Abstract. Key blinding produces pseudonymous digital identities by rerandomizing public keys of a digital signature scheme. It is used in anonymous networks to provide the seemingly contradictory goals of anonymity and authentication. Current key blinding schemes are based on the discrete log assumption. Eaton, Stebila and Stracovsky (LAT-INCRYPT 2021) proposed the first key blinding schemes from lattice assumptions. However, the large public keys and lack of QROM security means they are not ready to replace existing solutions.

We present a new way to build key blinding schemes form any MPCin-the-Head signature scheme. These schemes rely on well-studied symmetric cryptographic primitives and admit short public keys. We prove a general framework for constructing key blinding schemes and for proving their security in the quantum random oracle model (QROM).

We instantiate our framework with the recent AES-based Helium signature scheme (Kales and Zaverucha, 2022). Blinding Helium only adds a minor overhead to the signature and verification time. Both Helium and the aforementioned lattice-based key blinding schemes were only proven secure in the ROM. This makes our results the first QROM proof of Helium and the first fully quantum-safe public key blinding scheme.

1 Introduction

Public key signatures are used to verify the authenticity of messages by establishing that a valid signature must originate from the holder of the secret key. Message provenance and identities are thus closely tied to the signer's public key. This can be an issue in anonymized networks where the conflicting goals of authenticity and anonymity are required. Public key rerandomization [CGH+23] is one proposed solution to this conundrum. An example of rerandomization technique is public key blinding³, originally proposed in the Tor network's rendezvous specification for ECDSA. In signatures with public key blinding, the public key pk can be rerandomized using a seed τ into a blinded key bpk, such that knowledge of τ and bpk does not allow one to compute pk; and such that the secret key holder can produce valid signature for bpk with knowledge of τ .

³ Not to be confused with blind signatures, which allows signing a message while being oblivious to its content.

Post-quantum signature schemes can also admit key blinding [ESS21], but key blinding requires new security definitions and proofs of security that do not directly follow from the EUF-CMA security of the underlying scheme. In the case of post-quantum security, schemes that employ the random oracle methodology should be proven secure in the quantum random oracle model (QROM). Another challenge in constructing post-quantum signature schemes with key blinding is that most post-quantum signatures have large public keys, on the order of kilobytes for the Dilithium scheme selected for standardization by NIST. In the context of Tor's rendezvous spec, this is problematic since public keys represent identities that have to be handled manually by users.

In this work, we address all of these issues and more. We provide the first instantiation of post-quantum public key blinding with provable security in the QROM. Our scheme is based on the MPCitH methodology where signing involves proving knowledge of a preimage for a one-way function. Therefore our scheme has short public keys: they consist of a single element of the image of the OWF. We provide a generic construction for building key blinding schemes from any MPCitH-based signature scheme and establish a framework for proving their security in the QROM. We instantiate our construction with the Helium signature scheme, which is the latest in a long line of improvements to MPCitH schemes that boasts efficient and short proofs for statements about the AES block cipher. As a bonus contribution, we prove the security of the Helium signature scheme in the QROM, a question that was left open in [KZ22].

1.1 Our Results

We describe a framework for adding the public key blinding functionality to a broad family of signature schemes. We start from the idea of adding key blinding to the Picnic signature scheme which was sketched in [ESS21] and generalize it to any scheme where signatures consist of a message-dependent non-interactive zero-knowledge proof of knowledge (NIZKPoK) of the preimage of a one-way function constructed via Fiat-Shamir heuristic and the MPC-inthe-Head (MPCitH) paradigm⁴. We describe which properties those schemes, and the underlying pseudorandom function, must satisfy in order to yield an unforgeable and unlinkable key blinding scheme. The advantage of our modular approach is that signature with key blinding schemes can benefit from future improvements for MPCitH schemes. Our proofs are in the quantum-accessible random oracle model (QROM) where the adversary may evaluate the random oracle on an arbitrary quantum superposition of inputs. Since previous works only proved security against classical random oracle queries, this makes our construction the first fully quantum-safe signature scheme with key blinding.

We use the Helium scheme of [KZ22] to demonstrate our techniques. Helium offers several efficiency improvements over previous MPCitH-based schemes and

⁴ Although the MPCitH paradigm can be used to prove any NP statement, throughout this paper, we use the term to refer to proofs of statements about symmetric key primitives.

it is an appealing candidate since it is solely based on well-studied symmetric primitives such as the AES block cipher and the SHAKE extendable output function. Helium is the natural choice for proving statements about the AES circuit as it boasts the smallest proof sizes for the AES circuit, which has wellunderstood security. We apply our framework to the Helium signature scheme to get a signature scheme with key blinding called blHelium. To prove that our scheme is secure in the QROM, we prove that the proof system underlying Helium is a post-quantum proof of knowledge in the QROM. A direct corollary is that Helium is secure in the QROM, a question that was left open in [KZ22].

The blinded public keys for our blHelium scheme are only 32 bytes, which makes it an ideal candidate for a post-quantum transition from ECDSA for the Tor network and for other anonymity networks that require small keys.

We provide an open-source implementation⁵ of our blHelium scheme as a fork of the Helium code.

1.2 Related Work

Public key blinding was first introduced as an anonymity protection feature of Tor's Rendezvous protocol [Tor20]. Tor's key blinding scheme is based on discrete logarithm assumptions It also has applications to private airdrop and ratelimited privacy pass [ELW23]. Key blinding is just one of many public key rerandomization techniques. For an overview and comparison of signature schemes with key rerandomization, we refer the reader to [CGH+23].

The first post-quantum key blinding schemes were proposed in [ESS21]. Their constructions build on lattice-based post-quantum signature schemes and are only proven secure in the (classical) random oracle model. Being lattice-based, their schemes have fairly long public keys (on the order of kilobytes). The general construction of key blinding for MPCitH schemes that we present was first sketched in the appendix of [ESS21] for the Picnic signature scheme [CDG+17]. They briefly sketch a proof of unlinkability in the classical ROM, but provide no argument towards unforgeability.

Efforts have been made towards standardizing key blinding in an IETF technical specification draft [DELW23]. The companion paper [ELW23] provides security proofs for the scheme from the draft.

Multiparty computation in the head (MPCitH) is a technique for proving NP statements about arbitrary Boolean circuits introduced by [IKOS07]. This efficiency gains were introduced in ZKBoo [GMO16] and further improved in the paper that introduced Picnic [CDG+17], the first signature scheme built from the MPCitH framework to prove knowledge of a preimage of a OWF – in this case the LowMC block cipher which is optimized for low multiplicative complexity. Katz, Kolesnikov, and Wang [KKW18] added a preprocessing phase to the MPC computation used in [CDG+17]. BBQ [dSDOS20] is the first MPCitH signature scheme that instantiates the one-way function with the well studied

⁵ An anonymized repository is available at https://anonymous.4open.science/r/blinded-helium-aes-66E1/.

AES block cipher. It mitigates the larger circuit size by avoiding private keys that lead to circuits which have the 0 byte as input to an s-box, allowing for an efficient computation of the nonlinear operation. Baum and Nof introduces sacrificial multiplication triples (or beaver triples) to replace cut-and-choose checks, which leads to better soundness and less repetitions of the MPC protocol. The Banquet [BSK+21] signature scheme achieves 50% smaller signatures by running an MPC protocol that verifies the correctness of the circuit computation, instead of computing the result itself. Dobraunig, Kales, Rechberger, Schofnegger, and Zaverucha offers additional improvements to AES-based MPCitH signature schemes and proposes a scheme based on the *Rain* cipher optimized for MPCitH. Helium [KZ22] lifts multiple small fields \mathbb{F} elements into a larger field \mathbb{K} , such that multiplying the \mathbb{F} elements component-wise can be realised by a single operation on \mathbb{K} , a technique that was also used in Limbo [DOT21].

2 Preliminaries

For basic definitions on zero-knowledge proofs and proofs of knowledge, we refer to [Gol07].

2.1 Signature with Public Key Blinding

We definitions of this section are reproduced from previous work on signature schemes with key blinding [ESS21; ELW23].

Definition 1 (Digital Signature with Key Blinding). A digital signature scheme with key blinding scheme is a tuple of algorithms:

- KGen: returns a private key sk and an identity public key pk
- BlindPK(pk, τ): takes as input the identity public key pk and a blinding parameter τ and produces a blinded public key bpk_{τ}.
- Sign(sk, τ , m): produces a signature σ for m that is valid for the blinded key bpk_{τ} .
- Verify(bpk, m, σ): returns 1 if σ is a valid signature of m for blinded key bpk and 0 otherwise.

The scheme is (perfectly) correct if for $(sk, pk) \leftarrow KGen$ and $(bk, bpk) \leftarrow BlindPK(pk, \tau)$, then for all m and τ :

 $\mathsf{Verify}(\mathsf{bpk},m,\mathsf{Sign}(\mathsf{sk},\tau,m))=1$.

The unforgeability of signature schemes with key blinding is similar to that of regular unforgeability with the difference that we give the adversary control over which blinded key it targets for its forgery. The adversary is also allowed access to a signature oracle for an arbitrary (polynomial) number of blinded keys. **Definition 2 (Unforgeability – Chosen Message and Blinding Attack).** Let (KGen, BlindPK, Sign, Verify) be a key blinding signature scheme. The chosen message and blinding attack experiment EUF-CMBA is defined as the following game:

- The challenger samples $(pk, sk) \leftarrow KGen(1^n)$ and sends pk to A.
- \mathcal{A} can query the challenger on message m and blinding parameter τ to receive $\sigma = \text{Sign}(m, \text{sk}, \tau)$.
- \mathcal{A} sends its output (m^*, σ^*, τ^*) to the challenger who computes $bpk^* = BlindPK(pk, \tau^*)$ and outputs 1 if $Verify(bpk^*, m^*, \sigma^*) = 1$ and if (m^*, τ^*) was not previously queried to the signing oracle. Otherwise it outputs 0.

The advantage of an adversary \mathcal{A} for the EUF-CMBA game is defined as the probability $\operatorname{Adv}_{\mathcal{A}}^{\text{EUF-CMBA}}$ that the challenger outputs 1.

We define the existential unforgeability under blinded key only attack (EUF-BKO) as the same experiment, but where \mathcal{A} does not have access to a signing oracle (but can still compute blinded keys at will).

Note that a related work on key blinding [ELW23] allows signature queries and forgeries for the original public key pk. In our case, this is not desirable, since the signing algorithm will only produce signatures for blinded keys.

The notion of privacy provided by key blinding is that of unlinkability. A scheme is unlinkable if an adversary cannot tell if two blinded keys originate from the same identity public key or from different keys. In the unlinkability experiment, we also allow the adversary to request new blinded keys at will and to request signatures of arbitrary messages with respect to the blinded keys.

Definition 3 (Unlinkability – Chosen Message and Blinding Attack). Let (KGen, BlindPK, Sign, Verify) be a key blinding signature scheme. The unlinkability under chosen message and blinding attack (UL-CMBA) experiment is defined as the following game:

- The challenger samples $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{KGen}(1^n)$.
- \mathcal{A} can query a blinding oracle, which on input τ returns $\mathsf{bpk} \leftarrow \mathsf{BlindPK}(\mathsf{pk}_0, \tau)$.
- \mathcal{A} can query a signing oracle, which on a message m and a blinding parameter τ returns $\sigma = \text{Sign}(m, \text{sk}_0, \tau)$.
- \mathcal{A} sends a blinding parameter τ^* to the challenger. The challenger aborts the experiment if τ^* was previously queried to the blinding oracle.
- The challenger picks a new key pair $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KGen}(1^n)$, samples a bit $b \leftarrow \$ \{0, 1\}$ and sends $\mathsf{bpk}_b^* \leftarrow \mathsf{BlindPK}(\mathsf{pk}_b, \tau^*)$ to \mathcal{A} .
- \mathcal{A} again has access to the blinding and signing oracle, but now the oracles use the key pair $(\mathsf{sk}_b,\mathsf{pk}_b)$ if $\tau = \tau^*$ and defaults to the pair $(\mathsf{sk}_0,\mathsf{pk}_0)$ if $\tau \neq \tau^*$.
- \mathcal{A} outputs a guess b' and wins if b' = b.

The advantage of an adversary \mathcal{A} for the experiment is defined as the probability $\mathsf{Adv}_{\mathcal{A}}^{\text{UL-CMBA}} = \left|\Pr[b=b'] - \frac{1}{2}\right|.$

2.2 Quantum Random Oracle Model

In the quantum random oracle model (QROM), the adversary has quantum oracle access to a unitary $\mathcal{O}^H : |c\rangle|x\rangle|y\rangle \mapsto |c\rangle|x\rangle|y \oplus c \cdot H(x)\rangle$ that computes a random function H in superposition. While the QROM does not permit observing and reprogramming random oracle queries as easily as in the classical ROM, there are now powerful tools for proving security in the QROM, which we present below.

Theorem 1 (Measure-and-reprogram [DFM20; DFMS19]). Let \mathcal{X} and \mathcal{Y} be finite non-empty sets. There exists a black-box two-stage quantum algorithm \mathcal{S} with the following property. Let \mathcal{A} be an arbitrary oracle quantum algorithm that makes q queries to a uniformly random $H : \mathcal{X} \to \mathcal{Y}$ and that outputs some $x \in \mathcal{X}$ and a (possibly quantum) output z. Then, the two-stage algorithm $\mathcal{S}^{\mathcal{A}}$ outputs some $x \in \mathcal{X}$ in the first stage and, upon a random $\Theta \in \mathcal{Y}$ as input to the second stage, a (possibly quantum) output z, so that for any $x_{\circ} \in \mathcal{X}$ and any (possibly quantum) predicate V:

$$\begin{split} & \Pr_{\Theta} \Big[x = x_{\circ} \wedge V(x, \Theta, z) : (x, z) \leftarrow \langle \mathcal{S}^{\mathcal{A}}, \Theta \rangle \Big] \\ & \geq \frac{1}{(2q+1)^2} \Pr_{H} \Big[x = x_{\circ} \wedge V(x, H(x), z) : (x, z) \leftarrow \mathcal{A}^{H} \Big] \,. \end{split}$$

Furthermore, S runs in time polynomial in q, $\log |\mathcal{X}|$ and $\log |\mathcal{Y}|$.

Lemma 1 (One-Way to Hiding [Unr15]). Let $H : \mathcal{X} \to \mathcal{Y}$ be a quantumly accessible random oracle and let \mathcal{A}^H be an adversary that makes at most q queries to H. Let \mathcal{E}^H be an algorithm that picks $i \in [q]$ and $y \in \mathcal{Y}$ at random, runs $\mathcal{A}^H(x, y)$ until it's ith query, measures the input of the query in the computational basis and outputs the measurement outcome.

$$\left|\Pr[1 \leftarrow \mathcal{A}^{H}(x, H(x))] - \Pr[1 \leftarrow \mathcal{A}^{H}(x, y)]\right| \le 2q \cdot \sqrt{\Pr[x \leftarrow \mathcal{E}^{H}(x)]} \quad . \tag{1}$$

3 Key Blinding for MPC-in-the-Head Signature Schemes

3.1 Blinding MPCitH Public Keys

We refer to an MPCitH signature scheme as a signature schemes that respect the following structure⁶. Let $\mathcal{F} = \{f_k : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}\}_{k \in \{0,1\}^{\lambda}}$ be a family of pseudorandom functions. For $x \in \{0,1\}^{\lambda}$, we let $F_x : k \mapsto f_k(x)$. It was shown in the full version of [CDG+17] that F_x is a one-way function for any input x. For $\Pi = (\mathsf{P}, \mathsf{V})$, a ZKPoK for the relation $R = \{(y,k)\} \mid y = F_x(k)\}$, the MPCitH signature scheme proceeds as follows.

⁶ There are schemes based on the MPCitH paradigm for proving more general NP relations instead of statements about symmetric key primitives. We take the more narrow definition in this paper.

- Key generation: sample $x \in \{0,1\}^{\lambda}$ and $k \in \{0,1\}^{\lambda}$. Output $\mathsf{sk} := k$ and $\mathsf{pk} := (x, f_k(x))$.
- Signature: to sign a message m, use P on input $(\mathsf{pk},\mathsf{sk})$ to produce a noninteractive zero-knowledge proof of knowledge of k such that $\mathsf{pk} = (x, f_k(x))$ that depends⁷ on m.
- Verification: run the verification protocol V on input pk.

For such protocols, we consider a generic blinding procedure which was first informally proposed in [ESS21] and proceeds as follows. To blind a public key $\mathsf{pk} = (x, f_k(x))$ using a seed τ , one encrypts x a second time using a new key derived from pk and τ , for example using a cryptographic hash function H modelled as a random oracle. The new blinded public key is $\mathsf{bpk}_{\tau} = (x, f_{H(\tau,\mathsf{pk})}(\mathsf{pk}))$.

A few observations are in order. First, we want the same verification procedure for every blinded key (i.e., verification depends only on the blinded public key and does not require knowledge of τ), so the value x cannot be itself encrypted. Second, unlinkability requires that we use the same input x for every public key, otherwise it becomes trivial to link blinded keys to the original key. Based on these observations, we conclude that each public key must use the same input x.

Construction 1. Let $\mathcal{F} = \{f_k\}_{k \in \{0,1\}^{\lambda}}$ be a family of pseudorandom permutations, let inp be a fixed input and let $F^2(k, k') := f_{k'}(f_k(\text{inp}))$. Let $\Pi = (\mathsf{P}, \mathsf{V})$ be a ZKPoK for the relation $R = \{(y, (k, k')) \mid y = F^2(k, k')\}$. We define $\mathsf{blSig} = (\mathsf{KGen}, \mathsf{BlindPK}, \mathsf{Sign}, \mathsf{Verify})$ as the signature scheme with key blinding parameterized by \mathcal{F} and inp where

- KGen (1^{λ}) returns sk \leftarrow \$ $\{0,1\}^{\lambda}$ and pk = $f_{sk}(inp)$
- BlindPK(pk, τ) returns bpk = $f_{H(\tau, pk)}(pk)$
- Sign(sk, τ , m) use P on input (bpk, sk, $H(\tau, pk)$) to produce a message-dependent ZKPoK of a preimage of bpk for function F^2
- Verify(bpk, m, σ) returns 1 if σ is a valid message-dependent proof of knowledge of a preimage of bpk for F^2 .

We now present which conditions must be satisfied for Construction 1 to be a secure public key signature with key blinding scheme.

3.2 Unforgeability of Construction 1

We want to reduce the problem of forging valid signatures for blinded keys to that of inverting the one-way function $F^2: (k, k') \mapsto f_{k'}(f_k(x))$. Recall that for breaking the unforgeability game, the adversary on input pk must produce τ^* , m^* and a valid proof of knowledge σ^* of (k_1, k_2) such that $\mathsf{BlindPK}(\mathsf{pk}, \tau^*) = f_{H(\tau^*,\mathsf{pk})}(\mathsf{pk}) = f_{k_2}(f_{k_1}(x))$. In particular the adversary could produce such a proof for $(k_1, k_2) \neq (k, H(\tau^*, \mathsf{pk}))$, such that the knowledge extractor for the PoK

⁷ For example, if the proof system uses the Fiat-Shamir heuristic, m can be included in the random oracle queries that compute the verifier's challenge.

would not necessarily allow us recover the secret k. The following NP relation more precisely captures the problem the adversary needs to solve for forging signatures.

Definition 4. Let $R_{H,f}$ be the NP relation where the instances are of the form $y \in \text{Im}f$ and witnesses are tuples (k, k', τ) such that

$$(y,(k,k',\tau)) \in R_{H,f} \iff f_{H(\tau,y)}(y) = f_{k'}(f_k(x)) \quad . \tag{2}$$

We observe that the adversary does not have full control over the choice of k_2 in the unforgeability game: it is the output of the random oracle. We can use this fact to show that this relation is hard in the quantum random oracle model by reducing it to the hardness of inverting F^2 . We sketch the proof idea in the classical ROM and prove it below in the QROM.

Let \mathcal{A}^H be an adversary that on input y' outputs a witness for $R_{H,f}$ with some inverse polynomial probability. We can invert F^2 on a specific point y by sampling a key s, setting $y' = f_s^{-1}(y)$, running \mathcal{A}^H on input y', and reprogramming one of it's oracle queries to output s. With noticeable probability, \mathcal{A}^H outputs (τ, k, k') such that $f_{H(\tau, y')}(y') = f_{k'}(f_k(x))$ and again with noticeable probability, $H(\tau, y') = s$ so that $y = f_s(y') = f_{k'}(f_k(x))$ and (k, k') is a preimage of y for F^2 .

Lemma 2. Let $\{f_k\}_{k \in \{0,1\}^{\lambda}}$ be a family of pseudorandom permutations. If H is a quantum random oracle, then the hardness of the relation $R_{H,f}$ reduces to the problem of finding a preimage of $F^2: (k_1, k_2) \mapsto f_{k_1}(f_{k_2}(x))$.

Proof. We proceed as with the classical intuition, but use the measure-and-reprogram technique (Theorem 1) to insert s into a random oracle query. Let \mathcal{A}^H be an adversary against $R_{H,f}$ and let $\mathcal{S}^{\mathcal{A}}$ be the quantum algorithm from Theorem 1. The reduction \mathcal{R} proceeds as follows:

- 1. On input y, sample s uniformly at random and set $y' = f_s^{-1}(y)$.
- 2. Run the first stage of $\mathcal{S}^{\mathcal{A}}$ on input y' to get a point x.
- 3. Run the second stage of $\mathcal{S}^{\mathcal{A}}$ with input s.
- 4. When $\mathcal{S}^{\mathcal{A}}$ produces an output (τ, k, k') , output (k, k').

We now show that this reduction inverts F^2 with probability polynomially related to the probability that \mathcal{A}^H breaks relation $R_{H,f}$. Let V(s, k, k') be the predicate that returns 1 if and only if $f_s(y') = f_{k'}(f_k(x))$ such that $(y', (\tau, k, k')) \in$ $R_{H,f} \iff V(H(\tau, y'), k, k') = 1$. Let $x_0 = (\tau, y')$. Then by Theorem 1,

$$\begin{aligned} &\Pr[x = x_0 \land y = f_{k'}(f_k(x)) \mid (k,k') \leftarrow \mathcal{R}(y)] \\ &= \Pr[x = x_0 \land f_s(y') = f_{k'}(f_k(x)) \mid (k,k') \leftarrow \mathcal{R}(y)] \\ &= \Pr[x = x_0 \land V(s,k,k') = 1 \mid (k,k') \leftarrow \mathcal{R}(y)] \\ &= \Pr[x = x_0 \land V(s,k,k') = 1 \mid (\tau,k,k') \leftarrow \mathcal{S}^{\mathcal{A}}(y')] \\ &\geq \frac{1}{(2q+1)^2} \Pr[x = x_0 \land V(H(x),k,k') = 1 \mid (\tau,k,k') \leftarrow \mathcal{A}^{H}(y')] \end{aligned}$$

where q is the number of queries to H made by \mathcal{A}^H . By summing over x, the first term yields the advantage of \mathcal{R} for inverting F^2 and the last term is polynomially related to the advantage of \mathcal{A}^H for finding a witness for relation $R_{H,f}$.

We have reduced the hardness of $R_{H,f}$ to the one-wayness of F^2 . It is not hard to see that F^2 is one-way if f_k is pseudorandom. It follows from the observation that $\{f_{(k,k')}\}_{(k,k')}$ for $f_{(k,k')} = f_{k'} \circ f_k$ is a pseudorandom function family.

A useful tool for generically proving security of construction 1 is to be able to produce valid signature without the secret key by reprogramming the random oracle.

Definition 5 (Signature Simulation with Oracle Reprogramming). Let $\Pi = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a digital signature scheme (with or without key blinding) that relies on the random oracle H. Let \mathcal{G}^H be an arbitrary cryptographic game and let $\mathcal{A}^{H,\mathsf{Sig}}$ be a polynomial time adversary against game \mathcal{G}^H that makes $\mathsf{poly}(\lambda)$ quantum queries to H and $\mathsf{poly}(\lambda)$ signature queries. We say that Π is signature-simulatable in the QROM if there exists $\mathcal{B}^{H'}$, which plays game \mathcal{G} with a different random oracle H', and if there is a negligible function ϵ_{Sig} such that

$$\operatorname{\mathsf{Adv}}_{\mathcal{B}^{H'}}^{\mathcal{G}^{H'}} \ge \operatorname{\mathsf{Adv}}_{\mathcal{A}^{H},\operatorname{Sig}}^{\mathcal{G}^{H}} - \epsilon_{\operatorname{\mathsf{Sig}}}(\lambda) \quad . \tag{3}$$

Consider for example the unforgeability game. Adversary \mathcal{A} plays the chosen message unforgeability game and \mathcal{B} plays the no-message (key only) game. If \mathcal{A} produces a forgery using H and chosen message signatures, \mathcal{B} can produce a forgery by simulating the signature queries of \mathcal{A} through reprogramming its random oracle H. Since \mathcal{B} plays its game with a truly random oracle H', and message verification relies on H', the forgery produced by \mathcal{A} with the reprogrammed H should also be a valid signature with respect to H'.

Theorem 2. Let $\mathcal{F} = \{f_k\}_k$ be a family of pseudorandom permutations, let inp be a fixed input and let $F^2(k, k') := f_{k'}(f_k(\text{inp}))$. Let $\Pi = (\mathsf{P}, \mathsf{V})$ be a ZKPoK for the relation $R = \{(y, (k, k')) \mid y = F^2(k, k')\}$. Let $\text{Sig}_f = (\text{KGen}, \text{Sign}, \text{Verify})$ be the MPCitH signature scheme with key blinding from Construction 1. If Sig_f is ϵ_{Sig} -signature-simulatable in the QROM (Definition 5), then Sig_f is EUF-CMBA in the QROM with advantage at most

$$\mathsf{Adv}^{\mathrm{EUF-CMBA}} \le \mathsf{Adv}^{R_{H,f}} + \epsilon_{\mathsf{Sig}} + \epsilon_{KS}$$
 . (4)

Proof. Let $\mathcal{A}^{H,Sig}$ be an adversary against the EUF-CMBA of blHelium with access to a signature oracle Sign and a quantum random oracle H. By Definition 5, for any game \mathcal{G} in which $\mathcal{A}^{H,Sign}$ has oracle access to the signature algorithm of Π , the advantage of \mathcal{A} is polynomially related to the advantage of an adversary \mathcal{B}^{H} that does not have access to the Sign oracle. Therefore

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMBA}} \leq \mathsf{Adv}_{\mathcal{B}}^{\text{EUF-BKO}} + \epsilon_{\mathsf{Sig}}$$

Now, in the EUF-BKO game, \mathcal{B}^H must produce (m, σ, τ) where σ is a valid signature of m for the blinded key $\mathsf{bpk} = \mathsf{BlindPK}(\mathsf{pk}, \tau)$. The advantage of \mathcal{B}^H

in this game is

$$\mathsf{Adv}_{\mathcal{B}}^{\text{EUF-BKO}} = \Pr[\mathsf{Verify}(f_{H(\tau,\mathsf{pk})}(\mathsf{pk}), m, \sigma) = 1 \mid (m, \sigma, \tau) \leftarrow \mathcal{B}^{H}(\mathsf{pk})$$

By the construction of the signature scheme, σ is a proof of knowledge of (k, k') such that $f_{H(\tau,\mathsf{pk})}(\mathsf{pk}) = f_{k'}(f_k(\mathsf{inp}))$. Therefore, there exists an extractor \mathcal{E} such that

$$\begin{aligned} &\Pr[f_{H(\tau,\mathsf{pk})}(\mathsf{pk}) = f_{k'}(f_k(\mathsf{inp})) \mid (\tau, k, k') \leftarrow \mathcal{E}(\mathsf{pk})] \\ &\geq &\Pr[\mathsf{Verify}(f_{H(\tau,\mathsf{pk})}(\mathsf{pk}), m, \sigma) = 1 \mid (m, \sigma, \tau) \leftarrow \mathcal{B}^H(\mathsf{pk})] - \epsilon_{KS}(\lambda) \end{aligned}$$

where ϵ_{KS} is the knowledge soundness error of the interactive proof Π .

This extractor \mathcal{E} , on input pk, finds a witness for pk for the relation $R_{H,f}$. By Lemma 2, we have that the relation $R_{H,f}$ is hard in the QROM if f_k are pseudorandom functions, so it must be that the advantage $\mathsf{Adv}_{\mathcal{B}}^{\text{EUF-BKO}}$ is negligible. Concluding, we have that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{EUF-CMBA}} \le \mathsf{Adv}_{\mathcal{E}}^{R_{H,f}} + \epsilon_{\mathsf{Sig}} + \epsilon_{KS} \tag{5}$$



3.3 Unlinkability of Construction 1

We show unlinkability of Construction 1 in the QROM using a similar strategy as [ESS21]. That is, we first invoke signature simulation to remove the signature oracle. Next, we remove the dependence on the identity public key pk of the blinding secret key, computed as $H(\tau, pk)$, blinding with a random nonce instead. Then, we replace the blinding oracle to return encryptions of independently generated public keys. This reduces the unlinkability game to the task of distinguishing many encryptions of the same plaintext from many encryptions of independent plaintexts. This corresponds to the notion of chosen plaintext indistinguishability security in a multi-user setting.

Definition 6. Let (Enc, Dec) be a private-key encryption scheme. The multiuser indistinguishability experiment of parameter n is defined as follows:

- The challenger samples n keys $r_1, \ldots, r_n \leftarrow \{0, 1\}^{\lambda}$ and a bit $b \leftarrow \{0, 1\}$.
- The adversary may send queries of the form $(i, m_0, m_1) \in [n] \times \mathcal{M} \times \mathcal{M}$ to which the challenger responds to with $\mathsf{Enc}_{r_i}(m_b)$.
- The adversary outputs a bit b'.

The advantage of an adversary \mathcal{A} in the MU-IND game is $\operatorname{Adv}_{\mathcal{A}}^{\operatorname{MU-IND}}(\lambda) = |\operatorname{Pr}[b=b'] - \frac{1}{2}|.$

Multi-user security was studied in the context of public-key encryption in [BBM00] where the authors found that it is implied by IND-CPA up to some loss in security. Their results also apply to the private-key setting, thus if we assume that AES, our chosen cipher for blHelium, satisfies chosen plaintext indistinguishability, then it is secure according to Definition 6.

Theorem 3. Let $\{f_k\}_k$ be a MU-IND secure pseudorandom permutation family. If the blinded signature scheme Sig of Construction 1 is ϵ_{Sig} -signature-simulatable in the QROM, then Sig is unlinkable under chosen message and blinding attack (Definition 3) in the QROM with advantage

$$\mathsf{Adv}^{\text{UL-CMBA}} \le \frac{1}{2} + \mathsf{Adv}^{\text{MU-IND}} + 2Q_H \sqrt{\mathsf{Adv}^{\text{OWF}}} + 2\epsilon_{\mathsf{Sig}} \ . \tag{6}$$

Proof. We proceed by a game-hopping argument to reduce the unlinkability game to the multi-user indistinguishability of f_k . Game 0 is the UL-CMBA experiment. In each new game, we remove some part of the game that depends on the identity public key pk.

Game 1. Game 1 is as game 0, but where we remove oracle access to the signature oracle before and after the challenge. By assumption, the scheme admits signature simulation (Definition 5), therefore the success probability of $\mathcal{A}^{Sig,H}$ against game 0 is about the same of an adversary \mathcal{B}^{H} against game 1.

$$\left|\Pr[\mathcal{B}^{H} \text{ wins} \mid \text{Game 1}] - \Pr[\mathcal{A}^{\mathsf{Sig},H} \text{ wins} \mid \text{Game 0}]\right| \leq 2\epsilon_{\mathsf{Sig}}$$

where we doubled the simulation error ϵ_{Sig} since in game 0, the adversary effectively has access to two signing oracles (one for $\tau = \tau^*$ using key pk_b and one for $\tau \neq \tau^*$ using key pk_0) each of which needs to be simulated by \mathcal{B}^H .

Game 2. Next, we show how to remove dependence on pk from the symmetric key used to blind the public key. In game 2, whenever the adversary makes a query to the blinding oracle blind, the challenger uses a random value $k' \in \{0,1\}^{\lambda}$ and returns $f_{k'}(\mathsf{pk})$ instead of $f_{H(\tau,\mathsf{pk})}(\mathsf{pk})$. We may assume without loss of generality that the adversary queries the blind oracle at most once per input τ since the blind function is deterministic.

The only way the adversary can detect a change in the behavior of the blinding oracle is if it has queried H on an input containing pk. However, the adversary has quantum access to H so "having queried H on input pk" is ambiguous in this setting. We use the one-way to hiding Lemma [Unr15] to make this formal and to upper-bound the difference in both games with the probability of extracting pk from the blinding queries.

The one-way to hiding lemma allows us to relate the difference in winning probability between games 1 and 2 to the probability of extracting pk from the adversary's oracle queries. Let \mathcal{B}^H be the adversary against game 1 and let \mathcal{E}^H be an algorithm that runs \mathcal{B}^H , but that picks one of the quantum queries of \mathcal{B} to H at random, measures this query and outputs the result. Then the OW2H lemma states that

$$|\Pr[b = b' \mid \text{Game 2}] - \Pr[b = b' \mid \text{Game 1}]| \le 2Q_H \sqrt{\Pr[\mathsf{pk} \leftarrow \mathcal{E}^H]}$$
(7)

Since \mathcal{E} essentially runs \mathcal{B} and \mathcal{B} only sees encryptions of pk under different keys, the probability that \mathcal{E} outputs pk is at most the probability to recover the plaintext pk from many encryptions using different keys. We let $\mathsf{Adv}_{\mathcal{E}}^{\mathrm{OWF}} = \Pr[\mathsf{pk} \leftarrow \mathcal{E}^H]$ denote this negligible probability.

Game 3. We now change the blinding oracle again by replacing the real public key pk with a freshly sampled independent public key pk'. In game 3, when the adversary requests the blinding of the key with a seed τ , the challenger samples $k' \leftarrow \$ \{0,1\}^{\lambda}$ and $\mathsf{pk}' \leftarrow \mathsf{KGen}(1^{\lambda})$ and returns $f_{k'}(\mathsf{pk}')$.

We can relate the probability of success in this game with the multi-user indistinguishability of f_k . In game 3, the blinded keys that the adversary receives are encryptions of pk' under a secret key k' where both pk' and k' are unrelated to pk (and sk). If the adversary's behaviour changes in an observable way between games 4 and 3, then we can turn this into a distinguisher for the MU-IND property of f_k in the following way.

Let $\mathcal{C}^{\mathsf{blind}}$ be an adversary against game 2 with blinding oracle blind. Let \mathcal{D} be the following adversary against the MU-IND game:

- Let q be the number of blinding queries made by $\mathcal{C}^{\text{blind}}$. Sample q + 1 public keys $\mathsf{pk}_0, \ldots, \mathsf{pk}_n$ and query the MU-IND challenger on $(i, \mathsf{pk}_0, \mathsf{pk}_i)_{i \in [q]}$ to get the resulting ciphertexts c_1, \ldots, c_q (which are either an encryption of pk_0 or pk_i with a key k_i).
- \mathcal{D} now runs $\mathcal{C}^{\mathsf{blind}}$ by acting as the challenger in the UL-CMBA game and by simulating its blinding oracle as follows: on $\mathcal{C}^{\mathsf{blind}}$'s *i*th query to BlindPK, reply with c_i .
- Output whatever $\mathcal{C}^{\mathsf{blind}}$ outputs.

If b = 0 in the MU-IND game, then every c_i is an encryption of pk_0 and thus $\mathcal{C}^{\mathsf{blind}}$ is playing game 2. If b = 1, then each c_i is the encryption of a new public key pk_i , so $\mathcal{C}^{\mathsf{blind}}$ is playing game 3. By the MU-IND property of f_k , we have that

$$|\Pr[b = b' | \text{Game } 3] - \Pr[b = b' | \text{Game } 2]| \leq \mathsf{Adv}_{\mathcal{D}}^{\text{MU-INE}}$$

In game 3, the adversary has no signing nor blinding oracle and receives a public key pk_b^* from the challenger. Since pk_b^* has equal probability of being pk_0 or pk_1 , the advantage of \mathcal{C} in this game is exactly $\frac{1}{2}$. So we have

$$\mathsf{Adv}_{\mathcal{A}}^{\text{UL-CMBA}} \leq \frac{1}{2} + \mathsf{Adv}_{\mathcal{D}}^{\text{MU-IND}} + 2Q_H \sqrt{\mathsf{Adv}_{\mathcal{E}}^{OWF}} + 2\epsilon_{\mathsf{Sig}}$$
(8)

Assuming the multi-user indistinguishability of f_k , the above quantity is negligible.

4 The blHelium Signature Scheme with Key Blinding

We now present our **blHelium** protocol, which follows Construction 1 instantiated with the AES block cipher and the Helium proof system [KZ22]. We begin by giving a brief overview of the Helium signature scheme before describing the changes we make to equip it with key blinding, along with parameter choices. The detailed proof system can be found in Appendix A where we also prove the security of Helium in the QROM.

4.1 Overview of Helium

The scheme takes place over seven *phases*, representing the different message flows in the identification scheme prior to being converted to a signature scheme via the Fiat-Shamir transform (i.e., Phase 1 represents the first prover message, Phase 2 the first challenge, etc.). A high-level description of the proof system is as follows:

- Committing to MPC Party Seeds. Each MPC party's randomness is derived from a single seed which is committed to. The protocol runs through a distributed computation of AES, with each party holding a share of the secret key.
- Checking of the MPC Computation. Proving that the AES circuit was evaluated correctly means verifying correctness of the shares at each step of the MPC protocol. Every linear operation in the AES circuit can be evaluated locally. The non-linear S-box (a field inverse operation) is done efficiently by injecting shares of s and t such that $s \cdot t = 1$.
- Challenging the Checking Protocol. The injected shares of (s_i, t_i) must be checked for consistency. This is done efficiently by using polynomials Sand T interpolated such that $S(i) = s_i$ and $T(i) = t_i$. The prover distributes shares of $P = S \cdot T$ to the parties. To verify correctness of the polynomials, a test $P(R) = S(R) \cdot T(R)$ is performed for a random R.

Verification consists of reconstructing the view of each MPC party whose seed was opened, testing for consistency, and checking that the challenges were computed correctly.

4.2 The blHelium Signature Scheme

At a high level, the blHelium signature scheme with key blinding follows Construction 1. We modify the signature and verification protocols of Helium to accommodate signing and verification according to blinded keys. The most significant distinction with Helium is that our public and blinded keys consists of two ciphertexts instead of one. That is, we consider the family of one-way functions $(k_1, k_2) \mapsto (AES_{k_1}(x_1), AES_{k_2}(x_2))$ for fixed inputs x_1, x_2 which we set as the all-0 and the all-1 strings. We explain the rationale for this design choice in the next section. When blinding the public key, we again encrypt component-wise. In slightly more details, blHelium consists of the following algorithms:

- KGen: The secret key sk is selected at random from the set of keys such that the circuits for $AES_{sk}(\vec{0})$ and $AES_{sk}(\vec{1})$ have no s-Boxes which receives the 0 byte input. The public key consists of $pk = (AES_{sk}(\vec{0}), AES_{sk}(\vec{1}))$.
- BlindPK(pk, τ): computes bk $\leftarrow H(\tau, pk, t)$ and increments t until the circuits for $AES_{bk}(pk_0)$ and $AES_{bk}(pk_1)$ have no s-Boxes which receives the 0 byte input. The blinded public key is $bpk = (AES_{bk}(pk_0), AES_{bk}(pk_1))$.
- Sign(sk, τ , m): computes the blinding keys (bk, bpk) as in BlindPK and runs the Helium protocol to construct a message-dependent non-interactive ZKPoK of sk and bk such that bpk = (AES_{bk}(AES_{sk}($\vec{0}$)), AES_{bk}(AES_{sk}($\vec{1}$))).
- Verify(bpk, m, σ): runs the verification protocol for the proof of knowledge.

4.3 blHelium Parameters & Performance

In our configuration, we encrypt two different plaintexts (more on this below) in ECB mode, first with sk, and then with bk. The choice of plaintext is arbitrary but must be fixed for all users – for simplicity we have chosen the all 0 and all 1 plaintext. This requires a total of two AES128 key schedules (which takes $2 \times 40 = 80$ s-boxes) and four AES128 encryptions (which takes $4 \times 160 = 640$ s-boxes) for a total of 720 s-boxes. Recall that s-box computation in the MPC protocol are checked using polynomial interpolation. Since the degree of each polynomial is limited by the size of the field, we split these into 6 sets of polynomials $(S_i, T_i, P_i)_{i \in [6]}$, each being used to prove correctness of 720/6 = 120 s-box computations. In contrast, Helium requires two sets of polynomials.

The reason we encrypt two plaintexts (rather than Helium's one) is to account for the increased degree of flexibility induced by the blinding process. Recall that we reduce unforgeability under chosen message and blinding attack (EUF-CMBA) to the task of finding k, k' such that $bpk = f_{k'}(f_k(inp))$. We now look at the complexity of brute-force attacks for this task.

We first observe that Helium and blHelium only use a subset of possible keys, since we restrict to AES circuits which have no substitution box that receives the 0 byte as input. Using the methodology of [dSDOS20], we can estimate the fraction of keys for which the AES circuit has no 0-input s-box as $0.457 \approx (1-\frac{1}{256})^{200}$. Therefore, the number of admissible keys is approximately 2^{127} (i.e. the key space is reduced roughly by half).

Second, note that the brute force complexity of finding a preimage of **bpk** is at most $2 \cdot 2^{127}$ using a meet-in-the-middle attack, which was already remarked in [ESS21]. However, the adversary has even more flexibility at its disposal. That is because it is not restricted to mount an attack against a single plaintext and ciphertext pair. It choice of τ rerandomizes the public key. If we model a random τ as mapping the public key **pk** to a random blinded key **bpk**_{τ} and also assume that $AES_{bk'}(AES_{sk'}(x))$ is uniformly random when bk' and sk' are, then the adversary can pick triples (τ , bk', sk') at random until relation $R_{H,f}$ (eq. (2)) is satisfied. Since the ciphertext space is of size 2^{128} , the birthday bound implies that the adversary will find a good triple with constant probability after approximately $\sqrt{2^{128}} = 2^{64}$ attempts.

To retain 128 bits of security, we therefore increase the (blinded) public key space to 256 bits by encrypting two different plaintexts. Computing the public key now consists of one key schedule and two encryption circuits, both of which must have no 0-input s-Boxes. This requires 40 + 160 + 160 = 360 s-boxes, so we estimate that a fraction of around $(1 - \frac{1}{256})^{360} \approx 1/4$ of AES keys are valid (for a security loss of two bits in the secret key space). The same is true of the bk value when blinding keys: only around a quarter will be valid.

We provide an implementation of our protocol as a fork of the Helium implementation and report on the performance in Figure 1. Our focus in benchmarking is on signing and verification: key generation and blinding consist of only a handful AES operations and thus do not represent a significant burden. The sizes of public keys and signatures and the CPU time for signing and verification

are reported in Fig. 1 are compared to Helium. We can see that signature and verification are $2 \times$ to $3 \times$ slower than Helium and signature size are $2 \times$ to $3 \times$ larger. This is to be expected from the fact that the circuit form our blHelium scheme evaluates 720 s-boxes instead of 200 for Helium.

		Helium			blHelium	
(N, M)	Sign	Verify	Size	Sign	Verify	Size
(17, 31)	8.169	7.605	17580	16.816	14.086	52424
(19, 30)	8.088	7.507	17016	17.704	15.690	50736
(31, 26)	8.342	7.810	14760	20.095	17.569	43984
(57, 22)	9.918	9.370	12856	26.391	24.118	37584
(107, 19)	12.513	12.448	11420	38.459	35.607	32776
(139, 18)	14.267	14.196	11112	44.906	43.037	31344
(185, 17)	17.881	17.900	10500	55.109	52.827	29608
(255, 16)	21.636	21.593	9888	67.647	67.490	27872
(371, 15)	28.132	28.698	9516	90.303	88.953	26376

Fig. 1. Benchmarking (signature sizes in bytes and signing and verification times in milliseconds) information for our implementation of blinded Helium and comparison with Helium for different parameters N and M representing the number of MPC parties and the number of repetitions, respectively. Timing is averaged over 100 iterations.

4.4 Security of blHelium

The security of blHelium follows the general framework presented in Sections 3.2 and 3.3. Our security proofs for both unforgeability and unlinkability rely on the ability to simulate signatures using random oracle reprogramming, which we provide below for our blHelium scheme. Note that because of the similarities between blHelium and Helium, the same techniques apply to the Helium scheme through a minor modification to the scheme, namely the inclusion of the public key in the input to the oracles H_1 , H_2 and H_3 .

To simulate the signing oracle of blHelium, we can use techniques inspired by [KLS18] for their reduction of EUF-CMA to EUF-KO for Dilithium (a signature scheme also based on the Fiat-Shamir transform) where the random oracle is reprogrammed *offline* (prior to any query) using the HVZK simulator of the interactive proof, such that no recording of quantum queries is necessary. In [KLS18], they derandomize the simulator by making its random tape depend on the public key and the message to be signed. This allows the reprogrammed oracle to reply to queries on reprogrammed points consistently without the need to record those points.

In short, the proof of [KLS18] proceeds as follows. A signature for a message m consists of a transcript (w_m, c_m, z_m) of the non-interactive proof where $c_m = H(m, w_m)$. To simulate the signing oracle, the transcripts is instead produced by the HVZK simulator in a way that depends on m. The random oracle H

is programmed such that on input (m, w), it computes (w_m, c_m, z_m) using the simulator, checks if $w = w_m$ in which case it outputs c_m such that the transcript will be accepted by the verifier. To ensure that the transcript computed by the random oracle is the same as the one returned by the signing oracle, the simulator is derandomized and instead uses $\mathsf{RF}(m)$ as a random tape for a random function RF . To handle multiple signature queries, an additional input ctr is given to RF and incremented. The random oracle can then run through a list of transcripts to check if its input (m, w) is consistent with a simulated signature.

In our setting, the multi-round Fiat-Shamir means that we have to program multiple random oracles. We use a similar strategy where each oracle computes the simulator transcripts and is programmed to a specific challenge h'_i if one of the transcripts matches its input. To make this possible, we add m and the blinded public key **bpk** as input⁸ to every random oracle used to compute challenges (the Helium scheme only included them in the first oracle H_1). As in [KLS18], this programming is done ahead of any oracle query by the adversary and the programmed oracles can be evaluated in superposition. For this section, we assume that the oracle H_i are actually a single random oracle Hwith proper domain separation $H_i(x) = H(i, x)$. The reprogrammed oracles are described in Fig. 2.

Lemma 3 (Simulation of the Signature Oracle). For any cryptographic game \mathcal{G} , if an adversary $\mathcal{A}^{Sig,H}$ against blHelium with access to the signature oracle $Sig_{sk}(m,\tau)$ and to the quantum random oracle H has advantage $Adv_{\mathcal{A}^{Sig,H}}^{\mathcal{G}}$ in game \mathcal{G} , then there exists an adversary \mathcal{B}^{H} which has advantage

$$\mathsf{Adv}_{\mathcal{B}^H}^{\mathcal{G}} \geq \mathsf{Adv}_{\mathcal{A}^{\mathsf{Sig},H}}^{\mathcal{G}} - Q_s \cdot \epsilon_{zk} - \binom{Q_s}{2} \cdot 2^{-2\kappa}$$

where Q_s is an upper bound on the number of signature queries of \mathcal{A} and 2κ is the bit-length of salt in the Sig protocol.

Proof. The \mathcal{B}^H adversary will proceed by running $\mathcal{A}^{\operatorname{Sig},H}$ and whenever \mathcal{A} makes a signature query, it simulates the query by reprogramming the random oracle H using the HVZK simulator for the MPCitH interactive proof system. All other oracle queries to H are left unchanged. The reprogramming of the random oracle is done ahead of time, and so does not involve observing quantum queries (when the adversary makes a superposition query, the reprogrammed oracles are evaluated in superposition). Let Q_s be an upper-bound on the number of signature queries done by the adversary $\mathcal{A}^{\operatorname{Sig},H}$. Let $\operatorname{Sim}(\operatorname{bpk},r)$ be the deterministic $\operatorname{poly}(\lambda)$ -time machine that, on input bpk and random tape r produces a transcript $(\sigma_1, h_1, \sigma_2, h_2, \sigma_3, h_3, \sigma_4)$ for the 7-message protocol $\Pi_{\operatorname{blHelium}}$ that is indistinguishable from the real transcript. To make this change undetectable from the adversary's point of view, we need to reprogram the random oracles

⁸ Note that in [KLS18], the public key is not an input to the random oracles since it is fixed by the game. In our case, the adversary may issue signature queries for arbitrary blindings of the public key.

 $H_1(m, \mathsf{bpk}, \sigma_1)$

for $ctr = 1 \dots Q_s$, do 1: $(\sigma_1',h_1',\sigma_2',h_2',\sigma_3',h_3',\sigma_4') \gets \mathsf{Sim}(\mathsf{bpk},\mathsf{RF}(m,\mathsf{ctr}))$ 2:if $\sigma'_1 = \sigma_1$, return h'_1 3: return $H(1, m, \mathsf{bpk}, \sigma_1)$ 4: $H_2(m, \mathsf{bpk}, \mathsf{salt}, h_1, \sigma_2)$ for $ctr = 1 \dots Q_s$ do 1: $(\sigma'_1, h'_1, \sigma'_2, h'_2, \sigma'_3, h'_3, \sigma'_4) \leftarrow \mathsf{Sim}(\mathsf{bpk}, \mathsf{RF}(m, \mathsf{ctr}))$ 2:if salt' = salt $\wedge \sigma'_2 = \sigma_2 \wedge h'_1 = h_1$, return h'_2 3: return $H(2, m, bpk, salt, h_1, \sigma_2)$ 4: $H_3(m, \mathsf{bpk}, \mathsf{salt}, h_2, \sigma_3)$ 1:for $ctr = 1 \dots Q_s$, do $(\sigma'_1, h'_1, \sigma'_2, h'_2, \sigma'_3, h'_3, \sigma'_4) \leftarrow \mathsf{Sim}(\mathsf{bpk}, \mathsf{RF}(m, \mathsf{ctr}))$ 2:if salt' = salt $\wedge \sigma'_3 = \sigma_3 \wedge h'_2 = h_2$, return h'_3 3: return $H(3, m, bpk, salt, h_2, \sigma_3)$ 4:

Fig. 2. Reprogrammed random oracles for simulating the signature oracle of the blHelium signature scheme using oracle reprogramming. Sim is the HVZK simulator for the proof system and RF is a random function.

so that when H_i is queried on input σ_i , it returns h_i . Since the proof involves three such oracles H_1, H_2 and H_3 , we must reprogram each of them. We want to reprogram each oracle without having to record inputs, which might be in quantum superposition. We do this by having each oracle reproduce every simulated transcript and checking whether its input matches one of the transcripts. We derandomize the HVZK simulator so that each oracle can produce the same transcripts.

Let RF be a random function to which the adversary has no access. The machine \mathcal{B} computes the simulator's random tape as RF(m, ctr) where ctr is a counter that is incremented each time \mathcal{A} makes a signing queries. This ensures that Sim returns a fresh simulated transcript on each signing query. We reprogram H_i (for $i \in \{1, 2, 3\}$) as follows: iterate over $1 \leq ctr \leq Q_s$ and call the simulator with input bpk and random tape set to RF(m, ctr). If the transcript produced by Sim is consistent with the partial transcript in the oracle input, return the simulated challenge h'_i as output. If no consistent transcript was found, return the output of an unprogrammed oracle H_i . The detailed reprogrammed oracles are described in Figure 2.

Since \mathcal{B} replaces a real transcript with a simulated transcript, there is a probability ϵ_{zk} that $\mathcal{A}^{\text{Sig},H}$ observes this change⁹. Since there are at most Q_s signature queries, the probability that $\mathcal{A}^{\text{Sig},H}$ can distinguish from Q_s real from

⁹ See [KZ22, Appendix A] for a precise formulation of ϵ_{zk} for the BN++ signature scheme from which Helium is built.

 Q_s simulated transcripts is at most $Q_s \cdot \epsilon_{zk}$. The actual reprogramming of the random oracles is undetectable since for each signature query, a single point from each oracle H_i is reprogrammed to a uniformly and independently distributed value h_i .

Note that by reprogramming the random oracles in a correlated fashion as we do, there is a small chance that the output of the reprogrammed points is not consistent with the transcripts produced by the simulator. For example, if two transcripts start with the same σ_1 but differ afterwards, then oracle H_1 will return a value h_1 which, when fed in to H_2 will not return the right challenge. But since each transcript contains a random value salt (in σ_1) which is checked in each oracle query, the probability that the oracles answer inconsistently is at most the probability that two transcripts contain the same salt. It is at most $\binom{Q_s}{2} \cdot 2^{-2\kappa}$ since salt $\in \{0, 1\}^{2\kappa}$.

The advantage of \mathcal{B} in game \mathcal{G} is therefore at least

$$\mathsf{Adv}_{\mathcal{B}}^{\mathcal{G}} \geq \mathsf{Adv}_{\mathcal{A}^{\mathsf{Sig},H}}^{\mathcal{G}} - Q_s \cdot \epsilon_{zk} - \binom{Q_s}{2} \cdot 2^{-2\kappa}$$
.

In Appendix A, we prove the post-quantum security of the Helium proof system, i.e. we show that Helium is a post-quantum NIZKPoK of a preimage for a one-way function. Using this result, along with Lemma 3, we can apply Theorem 2 to obtain the following result.

Corollary 1. The blHelium signature scheme with key blinding is unforgeable under chosen message and blinding attack in the QROM with advantage

$$\mathsf{Adv}^{\text{EUF-CMBA}} \le \mathsf{Adv}^{R_{H,f}} + Q_s \cdot \epsilon_{zk} + \binom{Q_s}{2} \cdot 2^{-2\kappa} + \epsilon_{KS} \tag{9}$$

where ϵ_{KS} is given by (13) and ϵ_{zk} is the zero-knowledge error, and where Q_s is a bound on the number of signature queries.

Further assuming the multi-user indistinguishability of 128-bit AES, we have that the blHelium scheme is unlinkable.

Corollary 2. Assume that the AES block cipher is MU-IND (Definition 6), then blHelium is unlinkable under chosen message and blinding attack in the QROM with advantage

$$\mathsf{Adv}^{\text{UL-CMBA}} \le \frac{1}{2} + \mathsf{Adv}^{\text{MU-IND}} + 2Q_H \sqrt{\mathsf{Adv}^{\text{OWF}}} + 2Q_s \cdot \epsilon_{zk} + 2\binom{Q_s}{2} \cdot 2^{-2\kappa}$$
(10)

where ϵ_{zk} is the zero-knowledge error and where Q_s and Q_H are bounds on the number of signature and random oracle queries, respectively.

5 Conclusion

We have shown that MPCitH signature schemes based on symmetric cryptographic primitives are a prime candidate for key blinding. They only rely on well-studied assumptions and produce short public and blinded keys, which can be used as identifiers in anonymity networks. We have provided an implementation based on the Helium signature scheme that is ready to be experimented with.

Future work in this space would include adding the ability to *unblind* keys, which was presented in [ELW23] as *bidirectional* key blinding. For MPCitH-based schemes, it would suffice to recompute the secret key used to encrypt the public key and use it to decrypt the blinded key. Another question concerns strong unforgeability. In the context of quantum adversaries, strong unforgeability often relies on a concept known as *computationally unique responses*, which was used for example in [KLS18] and in [DFM20].

References

- [BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. "Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements". In: Advances in Cryptology — EUROCRYPT 2000. Ed. by Bart Preneel. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2000, pp. 259–274. ISBN: 978-3-540-45539-4. DOI: 10.1007/3-540-45539-6_18.
- [BN20] Carsten Baum and Ariel Nof. "Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography". In: Public-Key Cryptography - PKC 2020 -23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Proceedings, Part I. Vol. 12110. Lecture Notes in Computer Science. Springer, 2020, pp. 495–526.
- [BSK+21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. "Banquet: Short and Fast Signatures from AES". In: Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Proceedings, Part I. Vol. 12710. Lecture Notes in Computer Science. Springer, 2021, pp. 266-297.
- [CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. CCS '17. New York, NY, USA: Association for Computing Machinery, Oct. 30, 2017, pp. 1825–1842. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3133997. (Visited on 01/24/2023).

- [CGH+23] Sofía Celi, Scott Griffy, Lucjan Hanzlik, Octavio Perez Kempner, and Daniel Slamanig. Sok: Signatures With Randomizable Keys. 2023. URL: https://eprint.iacr.org/2023/1524 (visited on 04/16/2024). preprint.
- [DELW23] Frank Denis, Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. Key Blinding for Signature Schemes. Internet-Draft draftirtf-cfrg-signature-key-blinding-04. Work in Progress. Internet Engineering Task Force, July 2023. 15 pp. URL: https://datatracker. ietf.org/doc/draft-irtf-cfrg-signature-key-blinding/ 04/.
- [DFM20] Jelle Don, Serge Fehr, and Christian Majenz. "The Measure-and-Reprogram Technique 2.0: Multi-round Fiat-Shamir and More". In: Advances in Cryptology – CRYPTO 2020. Ed. by Daniele Micciancio and Thomas Ristenpart. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 602–631. ISBN: 978-3-030-56877-1. DOI: 10.1007/978-3-030-56877-1_21.
- [DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model". In: Advances in Cryptology – CRYPTO 2019. Ed. by Alexandra Boldyreva and Daniele Micciancio. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pp. 356–383. ISBN: 978-3-030-26951-7. DOI: 10.1007/978-3-030-26951-7_13.
- [DFMS22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. "Online-Extractability in the Quantum Random-Oracle Model". In: Advances in Cryptology – EUROCRYPT 2022. Ed. by Orr Dunkelman and Stefan Dziembowski. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 677– 706. ISBN: 978-3-031-07082-2. DOI: 10.1007/978-3-031-07082-2_24.
- [DKR+21] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. 2021. URL: https: //eprint.iacr.org/2021/692 (visited on 02/19/2024). preprint.
- [DOT21] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. "Limbo: Efficient Zero-knowledge MPCitH-based Arguments". In: CCS '21. New York, NY, USA: Association for Computing Machinery, Nov. 13, 2021, pp. 3022–3036. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3484595. (Visited on 05/01/2024).
- [dSDOS20] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. "BBQ: Using AES in Picnic Signatures". In: Selected Areas in Cryptography – SAC 2019. Ed. by Kenneth G. Paterson and Douglas Stebila. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 669–

692. ISBN: 978-3-030-38471-5. DOI: 10.1007/978-3-030-38471-5_27.

- [ELW23] Edward Eaton, Tancrède Lepoint, and Christopher A. Wood. Security Analysis of Signature Schemes with Key Blinding. Publication info: Preprint. 2023. URL: https://eprint.iacr.org/2023/380 (visited on 11/20/2023).
- [ESS21] Edward Eaton, Douglas Stebila, and Roy Stracovsky. "Post-quantum Key-Blinding for Authentication in Anonymity Networks". In: Progress in Cryptology – LATINCRYPT 2021. Ed. by Patrick Longa and Carla Ràfols. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 67–87. ISBN: 978-3-030-88238-9. DOI: 10.1007/978-3-030-88238-9_4.
- [GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. "ZKBoo: Faster Zero-Knowledge for Boolean Circuits". In: SEC'16. USA: USENIX Association, Aug. 10, 2016, pp. 1069–1083. ISBN: 978-1-931971-32-4.
- [Gol07] O. Goldreich. Foundations of Cryptography: Volume 1, Basic Tools. Cambridge University Press, 2007. ISBN: 978-1-139-43023-4.
- [IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.
 "Zero-Knowledge from Secure Multiparty Computation". In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing. STOC '07. New York, NY, USA: Association for Computing Machinery, June 11, 2007, pp. 21–30. ISBN: 978-1-59593-631-8. DOI: 10.1145/1250790.1250794. (Visited on 08/29/2023).
- [KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures". In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS '18. New York, NY, USA: Association for Computing Machinery, Oct. 15, 2018, pp. 525–537. ISBN: 978-1-4503-5693-0. DOI: 10.1145/3243734. 3243805. (Visited on 02/19/2024).
- [KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. "A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model". In: Advances in Cryptology: EUROCRYPT 2018. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 552–586. ISBN: 978-3-319-78372-7. DOI: 10.1007/978-3-319-78372-7_18.
- [KZ22] Daniel Kales and Greg Zaverucha. Efficient Lifting for Shorter Zero-Knowledge Proofs and Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2022/588. 2022. URL: https://eprint. iacr.org/2022/588.
- [Tor20] Tor Project. Tor Rendezvous Specification Version 3.0. 2020. URL: https://gitweb.torproject.org/torspec.git/tree/rendspec-v3.txt.

[Unr15] Dominique Unruh. "Revocable Quantum Timed-Release Encryption". In: Journal of the ACM 62.6 (Dec. 10, 2015), 49:1-49:76.
 ISSN: 0004-5411. DOI: 10.1145/2817206. URL: https://dl.acm.org/doi/10.1145/2817206 (visited on 05/07/2024).

A QROM Security of Helium

In the following, we refer to the "Helium proof system" as the 7-message interactive proof from which the Helium signature scheme is obtained through the Fiat-Shamir transform. We describe the prover's actions in the interactive proof in Figures 3 and 4. The verifier checks the validity of the commitments, recreates the views of the MPC parties using the seeds, and verifies that they are consistent with the MPC protocol. We refer to [KZ22] for the full details of the verifier. We let C denote the Boolean circuit that computes the one-way function.

A.1 Proof of Security

In this section, we prove the security of blHelium– i.e. we prove that it is unlinkable and unforgeable against an adversary that can request blinded keys and signatures at will. Our proofs hold in the quantum random oracle model (QROM). More specifically, we model the following hash functions as random oracles: commit, H_1 , H_2 and H_3 (in Sign) and H (in BlindPK).

Our proof relies on online extractability in the QROM. The following Theorem is the online extractability with early extraction result (Corollary 4 in [DFMS22] with the simplified bound from Theorem 3).

Theorem 4 (Online Extractability with Early Extraction [DFMS22]). Let $H : \mathcal{X} \to \mathcal{Y}$ be a random function. There exists an extractable RO-simulator \mathcal{S} , with interfaces $\mathcal{S}.RO$ and $\mathcal{S}.E$, that satisfies the following properties. Let \mathcal{A} be a two-round polynomial-time oracle adversary that outputs t_1, \ldots, t_ℓ in the first round and $x_1, \ldots, x_\ell \in \mathcal{X}$ and \mathcal{W} after the second round, resulting in a transcript $[\vec{t}, \vec{x}, H(\vec{x}), \mathcal{W}]_{\mathcal{A}^H}$. Let $[\vec{t}, \vec{x}, \vec{h}, \mathcal{W}]_{G_{\mathcal{S}}^{\mathcal{A}}}$ be the transcript where, when \mathcal{A} outputs t_i , $\mathcal{S}.E$ is queried on t_i to obtain $\hat{x}_i \in \mathcal{X} \cup \{\bot\}$ and when \mathcal{A} halts, $\mathcal{S}.RO$ is queried on \mathcal{A} 's outputs x_i to generate h_i . There are negligible functions δ_1 and δ_2 such that

$$\Pr_{\substack{G_{\mathcal{S}}\\G_{\mathcal{S}}}}[\exists i: x_i \neq \hat{x}_i \land h_i = t_i] \le \delta_1 \tag{11}$$

and

$$\Delta([\vec{t}, \vec{x}, H(\vec{x}), W]_{\mathcal{A}^H}, [\vec{t}, \vec{x}, \vec{h}, W]_{G^{\mathcal{A}}_{\mathfrak{s}}}) \le \delta_2$$

$$(12)$$

for $\delta_1 + \delta_2 \leq 34\ell q/\sqrt{2^n} + 2365q^3/2^n$ where q is the number of oracle queries.

Theorem 5. Assuming commit is a random oracle, then the Helium proof system instantiated with a one-way function F is a post-quantum proof of knowledge

First prover message Sample salt \leftarrow $\{0,1\}^{2\kappa}$ 1: for each parallel repetition $e \in [M]$ do 2:Sample root seed: seed_e \leftarrow \$ {0,1}^{κ} 3: Derive $\mathsf{seed}_e^{(1)}, \ldots, \mathsf{seed}_e^{(N)}$ as leaves of a binary tree from seed_e 4:for each party $i \in [N]$ do 5: Commit to seed: $\operatorname{com}_{e}^{(i)} \leftarrow \operatorname{Commit}(\operatorname{salt}, e, i, \operatorname{seed}_{e}^{(i)})$ 6: Expand random tape: tape⁽ⁱ⁾_e \leftarrow ExpandTape(salt, e, i, seed⁽ⁱ⁾_e) 7:Sample witness shares: $\mathsf{sk}_{e}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_{e}^{(i)}), \ \mathsf{bk}_{e}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_{e}^{(i)})$ 8: Compute witness offsets: $\Delta \mathsf{sk}_e \leftarrow \mathsf{sk} - \sum_i \mathsf{sk}_e^{(i)}, \ \Delta \mathsf{bk}_e \leftarrow \mathsf{bk} - \sum_i \mathsf{bk}_e^{(i)}$ 9: Adjust first shares: $\mathsf{sk}_e^{(0)} \leftarrow \mathsf{sk}_e^{(0)} + \Delta \mathsf{sk}_e$, $\mathsf{bk}_e^{(0)} \leftarrow \mathsf{bk}_e^{(0)} + \Delta \mathsf{bk}_e$ 10: 11: for each gate g in \mathcal{C} do if g is an addition with inputs x, y then 12:Party *i* locally computes $z^{(i)} = x^{(i)} + y^{(i)}$ 13:if g is a multiplication with inputs $x_{e,\ell}, y_{e,\ell}$ 14: Compute output share $z_{e,\ell}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$ 15: Compute offset $\Delta z_{e,\ell} \leftarrow x_{e,\ell} \cdot y_{e,\ell} - \sum_{i=1}^{N} z_{e,\ell}^{(i)}$ 16:Adjust first share $z_{e,\ell}^{(1)} \leftarrow z_{e,\ell}^{(1)} + \delta z_{e,\ell}$ 17: Let $\mathsf{ct}_e^{(i)}$ denote each party's output share. 18: for each party $i \in [N]$ do 19: $\text{Interpolate } S_e^{(i)} \text{ and } T_e^{(i)} \text{ such that } S_e^{(i)}(\ell) = s_{e,\ell}^{(i)} \text{ and } T_e^{(i)}(\ell) = t_{e,\ell}^{(i)} \text{ for } \ell \in [C]$ 20 : Compute product polynomial $P_e \leftarrow \left(\sum_i S_e^{(i)}\right) \left(\sum_i T_e^{(i)}\right)$ 21: for $\ell \in [C]$ do 22: Set $p_{e,\ell}^{(0)} \leftarrow 1$ and $p_{e,\ell}^{(i)} \leftarrow 0$ for $i \in \{1, \dots, N-1\}$ 23: for $\ell \in \{C, ..., 2C - 2\}$ do 24: $p_{e,\ell}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_{e}^{(i)}) \text{ for } i \in [N]$ 25:Compute offset $\Delta p_{e,\ell} = P_e(\ell) - \sum_i p_{e,\ell}^{(i)}$ 26: Adjust first share $p_{e,\ell}^{(0)} \leftarrow p_{e,\ell}^{(0)} + \Delta p_{e,\ell}$ 27: for each party $i \in [N]$ do 28: Interpolate polynomial $P_e^{(i)}(\ell) = p_{e,\ell}^{(i)}$ using the 2C-2 defined points. 29: $\sigma_1 \leftarrow (\mathsf{salt}, ((\mathsf{com}_e^{(i)}, \mathsf{ct}_e^{(i)})_{i \in [N]}, \Delta \mathsf{sk}_e, \Delta \mathsf{bk}_e, (\Delta t_{e,\ell})_{\ell \in [C]} (\Delta p_{e,\ell})_{\ell \in \{C, \dots, 2C-2\}})_{e \in [M]}).$ 30 :

Fig. 3. First prover message.

Second prover message on challenge h_1 1: Expand challenge: $(R_e)_{e \in [M]} \leftarrow \mathsf{Expand}(h_1)$, with $R_e \in \mathbb{K}$ for each repetition $e \in [M]$ do for each party $i \in [N]$ do 2: Sample shares $(a_e^{(i)}, c_e^{(i)}) \leftarrow \mathsf{Sample}(\mathsf{tape}_e^{(i)})$ 3: Compute $y_e = T_e(R_e)$ and $a_e = \sum_i a_e^{(i)}$ 4:Compute $\Delta c_e \leftarrow a_e \cdot y_e - \sum_i c_e^{(i)}$ 5:Adjust first share $c_e^{(0)} \leftarrow c_e^{(0)} + \Delta c_e$ 6: 7: Return $\sigma_2 \leftarrow (\Delta c_e)_{e \in [M]}$ Third prover message on challenge h_2 Expand challenge: $(\epsilon_e)_{e \in [M]} \leftarrow \mathsf{Expand}(h_2)$, with $\epsilon_e \in \mathbb{K}$ 1:for each repetition $e \in [M]$: 2: 3: for each party $i \in [N]$: Compute $(x_e^{(i)}, y_e^{(i)}, z_e^{(i)}) \leftarrow (S_e^{(i)}(R_e), T_e^{(i)}(R_e), P_e^{(i)}(R_e))$ 4: Compute $\alpha_e^{(i)} \leftarrow \epsilon_e \cdot x_e^{(i)} + a_e^{(i)}$ 5: Compute $\alpha_e \leftarrow \sum_i \alpha_e^{(i)}$ 6: for each party $i \in [N]$: 7: Compute $v_e^{(i)} \leftarrow \alpha_e \cdot y_e^{(i)} - \epsilon_e \cdot z_e^{(i)} - c_e^{(i)}$ 8: 9: Return $\sigma_3 \leftarrow (\alpha_e^{(i)}, v_e^{(i)})_{e \in [M], i \in [N]}$ Fourth prover message on challenge h_3 Expand challenge: $(\bar{i}_e)_{e \in [M]} \leftarrow \mathsf{Expand}(h_3)$ 1:for each repetition $e \in [M]$ do 2:seeds_e \leftarrow {the log₂(N) nodes needed to generate {seed_e⁽ⁱ⁾} for $i \in [N] \setminus \{\bar{i}_e\}$ } 3:4: Return (salt, h_1, h_3 , (seeds_e, com_e^($i_e)$, Δ sk_e, Δ bk_e, $(\Delta t_{e,\ell})_{\ell \in [C]}, (\Delta p_{e,\ell})_{\ell \in \{C, \dots, 2C-2\}}, \alpha_e, \Delta c_e)_{e \in [M]}$).</sup>

Fig. 4. Prover messages 2–4.

for preimages of F with knowledge soundness error

$$\epsilon_{KS} \le 34M \cdot N \cdot q/\sqrt{2^n} + 2365q^3/2^n$$

$$+ \max_{M_1, M_2, M_3} \left(\frac{2L-2}{|\mathbb{K}|}\right)^{M_1} \cdot \left(\frac{1}{2^8}\right)^{M_2} \cdot \left(\frac{1}{N}\right)^{M_3}$$
(13)

against quantum polynomial-time adversaries making q queries to commit where n is the bit size of commitments and where $M_1 + M_2 + M_3 = M$.

Proof. A quantum adversary \mathcal{A} against Helium has quantum superposition access to the commitment oracle $H_c = \text{commit.}$ We need to construct a knowledge extractor \mathcal{E} whose success probability in producing a witness is related to \mathcal{A} 's probability of cheating the protocol. The extractor will simulate the random oracle H_c with the online extractable oracle S of [DFMS22] specified in Theorem 4. The extractor will run \mathcal{A} by replacing H_c with the oracle interface $\mathcal{S}.RO$. After the prover's first message σ_1 , \mathcal{E} will use the extraction interface $\mathcal{S}.E$ on every commitment $\mathsf{com}_{e}^{(i)}$ for $e \in [M]$ and $i \in [N]$ to get either an oracle input $(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)})$ or a symbol \perp which means that \mathcal{A} did not query H_c on this input. Then, \mathcal{E} proceeds in a similar fashion as the extractor of [KZ22, Appendix A] in their proof of unforgeability of BN++ to compute the inputs and shares of the MPC parties.

In more details, when \mathcal{A} produces its first message σ_1 , \mathcal{E} does the following:

- parse σ_1 as (salt, $((\mathsf{com}_e^{(i)}, \mathsf{ct}_e^{(i)})_{i \in [N]}, \ldots)$,
- for $i \in [N]$ and $e \in [M]$, use the S.E interface on input $\operatorname{com}_{e}^{(i)}$ to get either
- $\hat{s}_e^{(i)}$ or \perp . For each $i \in [N]$ and $e \in [M]$ such that $\mathsf{seed}_e^{(i)}$ is successfully extracted, compute the input shares $\mathsf{sk}_e^{(i)}$ of party *i* for repetition *e* using the seed contained in $\hat{s}_e^{(i)}$.
- If there is an $e \in [M]$ for which $\mathsf{sk}_e = \sum_i \mathsf{sk}_e^{(i)}$ is a preimage for pk , output sk_e . Otherwise, output \perp .

We now show that the probability that $\mathcal E$ outputs a preimage sk is non-negligible if \mathcal{A} produces a forgery with non-negligible probability.

We let $V(\vec{\sigma}, \vec{h})$ denote the probabilistic event that V accepts the transcript $(\vec{\sigma}, \vec{h})$. We denote by \vec{s}_{h_3} the set of seeds announced when the challenge is $h_3 =$ $(\overline{i}_e)_{e \in [M]}$, i.e. $\vec{s}_{h_3} = \{(\mathsf{salt}, e, i, \mathsf{seed}_e^{(i)}) \mid e \in [M], i \neq \overline{i}_e\}$. Similarly, let \vec{y}_{h_3} denote the commitments to the revealed seeds; i.e. $\vec{y}_{h_3} = H_c(\vec{s}_{h_3})$.

We first bound the difference in probability between an execution with H_c and an execution with $\mathcal{S}.RO$.

$$\begin{aligned} \Pr[V(\vec{\sigma}, \vec{h})] &= \Pr[V(\vec{\sigma}, \vec{h}) \wedge H_c(\vec{s}_{h_3}) = \vec{y}_{h_3}] \\ &\leq \Pr[V(\vec{\sigma}, \vec{h}) \wedge \mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3}] \\ &+ \Pr[H_c(\vec{s}_{h_3}) = \vec{y}_{h_3} \wedge H_c(\vec{s}_{h_3}) \neq \mathcal{S}.RO(\vec{s}_{h_3})] \\ &\leq \Pr[V(\vec{\sigma}, \vec{h}) \wedge \mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3}] + \delta_1 \end{aligned}$$

where δ_1 is the negligible error term of (11).

Next, we bound the probability that the values $\hat{s}_e^{(i)}$ obtained through the $\mathcal{S}.E$ interface differ from the committed values.

$$\begin{aligned} &\Pr[V(\vec{\sigma}, \vec{h}) \land \mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3}] \\ &\leq \Pr[V(\vec{\sigma}, \vec{h}) \land \mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3} \land \mathcal{S}.E(\vec{y}_{h_3}) = \vec{s}_{h_3}] \\ &+ \Pr[\mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3} \land \mathcal{S}.E(\vec{y}_{h_3}) \neq \vec{s}_{h_3}] \\ &\leq \Pr[V(\vec{\sigma}, \vec{h}) \land \mathcal{S}.RO(\vec{s}_{h_3}) = \vec{y}_{h_3} \land \mathcal{S}.E(\vec{y}_{h_3}) = \vec{s}_{h_3}] + \delta_2 \end{aligned}$$

where $\hat{s}_{h_3} = \mathcal{S}.E(\vec{y}_{h_3})$. We again take note of δ_2 and add it at the end.

The expression $\Pr[V(\vec{\sigma}, \vec{h}) \wedge \hat{s}_{h_3} = \vec{s}_{h_3}]$ corresponds to the probability that V accepts when the committed seeds are the values $\hat{s}_e^{(i)}$ extracted by \mathcal{E} through $\mathcal{S}.E$. Note that the prover sends $N \cdot M$ commitments $y_e^{(i)}$, so the values $\hat{s}_e^{(i)}$ are well defined for each $(i, e) \in [N] \times [M]$. We let $\mathsf{sk} \leftarrow \mathcal{E}$ denote the event that the seeds $\hat{s}_e^{(i)}$ allow \mathcal{E} to compute the shares of the witness sk and $\perp \leftarrow \mathcal{E}$ its complement (when \mathcal{E} outputs \perp). We have

$$\begin{split} &\Pr[V(\vec{\sigma},\vec{h}) \wedge \hat{s}_{h_3} = \vec{s}_{h_3}] \leq \\ &\Pr[\mathsf{sk} \leftarrow \mathcal{E}] + \Pr[V(\vec{\sigma},\vec{h}) \wedge \hat{s}_{h_3} = \vec{s}_{h_3} \mid \bot \leftarrow \mathcal{E}] \enspace . \end{split}$$

We now look at this last probability that the verifier accepts when the extractor is unable to reconstruct a witness from the MPC shares.

Recall that the verifier recomputes the view of each party in the MPC protocol to the exception of the excluded party \bar{i} . Since $\hat{s}_{h_3} = \vec{s}_{h_3}$, it computes those views using the extracted seeds $\hat{s}_e^{(i)}$. If \mathcal{E} outputs \perp , then for every e it holds that the key $\hat{s}_e = \sum_i \hat{s}_e^{(i)}$ expanded from the seed $\hat{s}_e^{(i)}$ is not a valid preimage. This means that at least one of the views computed by the verifier is inconsistent (i.e. that party cheated). In this case the verifier accepts if one of three things happen:

1. the prover injects invalid polynomials S, T and P (such that $S \cdot T \neq P$); or

2. the prover injects invalid multiplication triples; or

3. the view of the inconsistent party is not opened.

There are M parallel repetitions which must pass verification and the prover may try to cheat in a different round in each repetition. We analyze the probability of each event below. The probability that the prover cheats in all M repetitions corresponds to the *trivial* cheating probability in the proof of [DFMS22]. At this point, the rest of the analysis is entirely classical and is very similar to the proof of [KZ22] and to other proofs of soundness for multi-round interactive proofs. We bound the probability that the verifier accepts when the MPC shares are computed using the extracted seeds $\hat{s}_e^{(i)}$, conditioned on the extraction failing. Cheating the first challenge. The first challenge h_1 is used to test the checking polynomials. In Helium, there are two checking polynomials P_1 and P_2 since there are not enough field elements in \mathbb{F}_{2^8} to interpolate a single polynomial with the desired degree. For other one-way functions, there might be more polynomials, for example in blHelium, we use a total of 6 checking polynomials since we are effectively applying 4 AES circuits (see Section 4.3 for details). Let n_p denote the number of checking polynomials and C the total count of field inverse in the circuit such that the degree of each polynomial is $L = \lceil C/n_p \rceil$. By the Schwartz–Zippel Lemma, the probability that a random point R_e satisfies $S_e(R_e) \cdot T_e(R_e) - P_e(R_e) = 0$ is at most $\frac{2L-2}{|\mathbb{K}|}$ where \mathbb{K} is the extension field of \mathbb{F}_{2^8} . The challenge h_1 is parsed as $(R_e)_{e \in [M]}$ where for each e, R_e is used to check that $S \cdot T = P$ by checking that $S(R_e) \cdot T(R_e) = P(R_e)$. If we let M_1 denote the number of parallel repetitions e for which the prover cheats in round 1, the probability that the adversary isn't caught is at most

$$\left(\frac{2L-2}{|\mathbb{K}|}\right)^{M_1} . \tag{14}$$

Cheating the second challenge. The second challenge h_2 is used to challenge the multiplication triples used to check $S_e(R_e) \cdot T_e(R_e) = P_e(R_e)$. The Helium protocol uses a dot-product checking protocol, which has soundness $1/|\mathbb{F}_{2^8}|$. If we let M_2 denote the number of repetitions where the adversary cheats in the second round, its probability of passing verification is at most

$$\left(\frac{1}{2^8}\right)^{M_2}$$

Cheating the third challenge. The third challenge is used to challenge the views of the MPC protocol. If the prover did not cheat in any of the previous two rounds, then there is at least one party whose view is inconsistent with that of the others. The prover can cheat in this round if the inconsistent view is not challenged by the verifier. This occurs with probability $\frac{1}{N}$. There are $M_3 = M - M_1 - M_2$ repetitions where the adversary attempts to cheat in the last round. So the probability of success in this round is

$$\left(\frac{1}{N}\right)^{M_3}\tag{15}$$

To complete the proof, we add all the error terms and obtain the bound

$$\begin{aligned} &\Pr[V(\vec{\sigma},\vec{h})=1] \leq \Pr[\mathsf{sk} \leftarrow \mathcal{E}] + \delta_1 + \delta_2 \\ &+ \max_{M_1,M_2,M_3} \left(\frac{2L-2}{|\mathbb{K}|}\right)^{M_1} \cdot \left(\frac{1}{|\mathbb{F}_{2^8}|}\right)^{M_2} \cdot \left(\frac{1}{N}\right)^{M_3} \end{aligned}$$

where the bound $\delta_1 + \delta_2 \leq 34\ell q/\sqrt{2^n} + 2365q^3/2^n$ is given by Theorem 4. Since our extractor extracts $\ell = M \cdot N$ points, the Theorem statement follows. \Box

After establishing the knowledge soundness of Helium against quantum provers, the rest of the proof towards unforgeability follows the usual formula. Theorem 5 together with the multi-round security of the Fiat-Shamir transform in the QROM [DFM20] directly imply that the Helium signature scheme is a non-interactive post-quantum zero-knowledge proof of knowledge. Therefore it is unforgeable under key-only attack. We have shown in Lemma 3 that the signature oracle for the blHelium signature scheme (which is just the Helium scheme with a different one-way function) can be simulated with random oracle reprogramming. A direct consequence is that the EUF-CMA of Helium reduces to EUF-KOA in the QROM.

Corollary 3. The Helium signature scheme is EUF-CMA in the QROM.