A compressed version of this paper appears in the proceedings of Latincrypt 2021. This is the full version.

Post-Quantum Key-Blinding for Authentication in Anonymity Networks

Edward Eaton, Douglas Stebila, and Roy Stracovsky

University of Waterloo
{eeaton, dstebila, rstracovsky}@uwaterloo.ca

Abstract. Anonymity networks, such as the Tor network, are highly decentralized and make heavy use of ephemeral identities. Both of these characteristics run in direct opposition to a traditional public key infrastructure, so entity authentication in an anonymity network can be a challenge. One system that Tor relies on is key-blinded signatures, which allow public keys to be transformed so that authentication is still possible, but the identity public key is masked. This is used in Tor during onion service descriptor lookup, in which a `.onion` address is resolved to a rendezvous point through which a client and an onion service can communicate. The mechanism currently used is based on elliptic curve signatures, so a post-quantum replacement will be needed.
We consider four fully post-quantum key-blinding schemes, and prove the unlinkability and unforgeability of all schemes in the random-oracle model. We provide a generic framework for proving unlinkability of key-blinded schemes by reducing to two properties, signing with oracle reprogramming and independent blinding. Of the four schemes, two are based on Round 3 candidates in NIST's post-quantum signature standardization process, Dilithium and Picnic. The other two are based on much newer schemes, CSI-FiSh and LegRoast, which have more favourable characteristics for blinding. CSI-FiSh is based on isogenies and boasts a very small public key plus signature sizes, and its group action structure allows for key-blinding in a straightforward way. LegRoast uses the Picnic framework, but with the Legendre symbol PRF as a symmetric primitive, the homomorphic properties of which can be exploited to blind public keys in a novel way. Our schemes require at most small changes to parameters, and are generally almost as fast as their unblinded counterparts, except for blinded Picnic, for which signing and verifying is roughly half as fast.

1 Introduction

Among the many difficulties in building a robust anonymity network, how entities are authenticated can be a unique challenge that cannot be solved with direct cryptographic techniques. Most networks will accomplish authenticity goals

through the use of a signature scheme, but in a network with anonymity goals, the public keys used for signing can run contrary to those goals. One technique to overcome this contradiction is used in the Tor network: *key-blinding*.

A signature scheme with key-blinding works similarly to a regular signature scheme, but with the added property that given a public key pk and a nonce τ, a new public key pk_τ can be derived, which in turn can be used for signing and verification. This is useful in contexts where two parties wish to exchange signed material, but must do so in the presence of a potential eavesdropper who may attempt to de-anonymize them. Tor describes such a scheme, and its use, in version 3 of the rendezvous specification, describing how clients connect to onion services in the network [35]. In subsection 2.1 we will describe precisely how key-blinding is used in Tor, and the security it is meant to provide.

It is useful to describe the key-blinding scheme as it exists in Tor, to gain some intuition for how such a scheme works and what security it provides. Key-blinding in Tor today uses the Ed25519 signature scheme [8]. Keys in this signature scheme are made with respect to a generator B of a cyclic group of size ℓ (written with additive notation). Secret keys are an integer $a \in \{1, \ldots, \ell - 1\}$ and the corresponding public key is $A = aB$. We refer to [8] for a complete description of the signing and verification processes, but for our description, it suffices to know that any such (a, A) pair are a valid key pair for Ed25519.

To blind a public key with a nonce τ, one computes a value $t \leftarrow H(\tau||A)$, with $t \in \{1, \ldots, \ell - 1\}$. Then the blinded public key is tA, with corresponding secret key $t \cdot a \pmod{\ell}$. This forms a new key pair that is entirely compatible with Ed25519, so that it can be used for signing and verification.

It is fairly easy to see why this scheme has the desired security properties. Given two blinded keys and the associated nonces, there is no way to tell if they come from the same identity public key or not. Without knowledge of the identity public key, the distribution of the blinded public key is entirely uniform over the public key space, so that these keys are entirely unlinkable to each other. Furthermore, the keys retain their unforgeability, as the blinded secret key ta requires both t and a to be known. Formal proofs of the security properties can be found in a tech report posted to the Tor developer mailing list [24].

This system works quite well for Tor today, but with the development of quantum computers, cryptography based on the discrete logarithm problem will eventually be rendered insecure. To ensure the long-term security of Tor, a replacement post-quantum signature scheme with key-blinding will be needed.

1.1 Our Contributions and Paper Structure

In our work we address the challenge of extending post-quantum signature schemes to have a key-blinding functionality. We focus on three promising post-quantum signature schemes. *Dilithium* is a lattice-based signature scheme that is currently under consideration in NIST's Post-Quantum Cryptography standardization effort [19]. Instead of directly working with Dilithium, we will work with the Dilithium-QROM variant [27]. Dilithium-QROM has simpler provable guarantees by neatly fitting into the 'Lossy ID scheme' framework [1], so to ensure that our

Scheme	$\|pk\|$	$\|\sigma\|$	KeyGen	Blind	Sign	Verify
Dilithium-QROM	7.7 kB	5.7 kB	3810 ms	-	9360 ms	2890 ms
blDlithium-QROM	10 kB	5.7 kB	2180 ms	1650 ms	28300 ms	717 ms
Increase from blinding	1.3×	1×	0.6×	-	3×	0.25×
LegRoast	0.50 kB	7.94 kB	0.9 ms	-	12.4 ms	11.7 ms
blLegRoast	0.50 kB	11.22 kB	0.9 ms	0.9 ms	18.6 ms	17.8 ms
Increase from blinding	1.0×	1.4×	1.0×	-	1.5×	1.5×
CSI-FiSh-Merkleized	32 B	1.8–2.1 kB	10900 ms	-	559 ms	559 ms
CSI-FiSh-unMerkleized	16 kB	0.45 kB	10800 ms	-	554 ms	553 ms
blCSI-FiSh	16 kB	0.45 kB	10600 ms	10600 ms	546 ms	540 ms
Increase from blinding	1.0×	1.0×	1.0×	-	1.0×	1.0×

Fig. 1. Performance results from the implemented key-blinding schemes. Note that we emphasize the increase over the raw numbers for the timing information. Implementations are not optimized and may not reflect how long a 'proper' implementation will take. Nonetheless, the increase reflects how much additional work is required to use the scheme for key-blinding. For all schemes, blinded public keys have the same size as their unblinded version and so we do not distinguish between the two.

scheme has similar guarantees we work within the same framework. *CSI-FiSh* is a relatively new post-quantum signature scheme based on the CSIDH group action [14]. *LegRoast* is based on the Picnic framework, but replaces the more traditional symmetric function used with the Legendre PRF, the homomorphic properties of which allow for small signatures [12, 17]. In the Appendix, we also consider Picnic itself. *Picnic* is another submission to NIST's efforts, which constructs a signature scheme out of the 'MPC-in-the-head' paradigm [16, 25]. We show how all of these signature schemes can be extended to support key-blinding. For CSI-FiSh, Dilithium, and LegRoast this process is done similarly to the existing Ed25519 scheme, by homomorphically incorporating a blinding factor into the public key. In all the schemes, blinding is generally around as efficient as key generation, while signing is either as efficient, or at worst half as fast. We provide a generic framework for proving the unlinkability property, showing that it reduces to two easily proven properties. We prove all these schemes both unlinkable and unforgeable in the random oracle model. Note that each of the signature schemes we have discussed are built out of the Fiat-Shamir paradigm. We discuss why this is the case, and what some of the challenges are for building a key-blinded scheme out of a trapdoor signature scheme. Finally, we provide prototype implementations out of the CSI-FiSh, LegRoast, and Dilithium-QROM schemes and discuss aspects of their performance as it applies to Tor. Our results from the three implemented schemes are shown in Figure 1.

Section 2 provides background information on Tor and definitions for key-blinding, which is needed to follow discussion in the remainder of the paper. We discuss the property of unlinkability, and how it can be achieved in all of our schemes in Section 3. In Section 4 we extend Dilithium-QROM with

key-blinding and discuss the proof of security, then repeat this process with CSI-FiSh in Section 5, and LegRoast in Section 6. Finally we provide details of our implementations and their performance in Section 7 before concluding in Section 8.

In the auxiliary material provided, we include an outline of a key-blinded Picnic scheme (Appendix B), as well as the detailed descriptions of the other schemes and the associated proofs of unlinkability and unforgeability (Appendices A, C, D, and E).

1.2 Related Work

As mentioned, the schemes that we choose to base blinded signature schemes off of are Dilithium [19,27], CSI-FiSh [14], LegRoast / PorcRoast [12], and, in the appendix, Picnic [16]. While to our knowledge, this is the first attempt to construct post-quantum key-blinded signatures, there are a few other papers who have attempted to build similar primitives, for different reasons. In a 2018 preprint [3], the authors considered post-quantum PKIs in vehicle-to-anything (V2X) communications. One of the techniques they developed to provide anonymity to vehicles in such a context involved transformations on public key materials similar to that of key-blinding. Their construction was based on the lattice scheme qTESLA, which was a candidate for standardisation in the first two rounds of NIST's process. The process of key-blinding also bears a similarity to hierarchical deterministic wallets used for Bitcoin [22,29]. These wallets allow a user to create child public and secret keys for the delegation of abilities for spending. Such a protocol has much stronger requirements than simple key-blinding, which does not need to be hierarchical and does not need for the child secret keys to contain no information about parent secret keys. Some work on post-quantum deterministic wallets has been published [2], with their scheme also based on qTESLA.

It is important to distinguish between the key-blinding schemes we discuss here and the notion of 'blind signatures', for which post-quantum schemes already exist [23,31]. Blind signatures are an interactive protocol that allow a user to obtain a signature on a message without the signer knowing the message. This is very different from key-blinded schemes, which have the same functionality as a traditional signature scheme, but with the extra ability to randomize public keys.

2 Background

2.1 Onion Services

The Tor network serves millions of clients a day, providing anonymity to users from the websites they connect to, and concealing what they are connecting to from their Internet service provider and any other intermediary in their path [34]. An important part of the Tor networks is *onion services* (previously known as hidden services). Onion services allow users to not only *access* content with Tor's strong privacy guarantees, but also *serve* content.

At a high level, onion services work by uploading a three hop path (called a *circuit* in Tor terminology) to a Tor node called a *introduction point*. This path begins at the introduction point and ends at the onion service. Because of Tor's layered encryption, the introduction point does not know where the onion service lives, only where the next node in the path lives. For a client to connect to the onion service, they use the `.onion` address to find the introduction point, who will then direct their communication towards the onion service.

In the most recent version of the rendezvous specification (the specification that describes the process of connecting to an onion service), the `.onion` address is the long-lived EdDSA public key of the onion service. Time in the Tor network is divided into periods, with the period length a consensus parameter and the period number the number of periods that have occurred since the Unix epoch. So given a public key, a nonce, and consensus parameters of the Tor network, the *blinding factor* t is computed by hashing together the public key, the nonce, and the current period number, as well as some parameters of both the Tor network and the signature scheme. As mentioned in the introduction, this value t is treated as an integer in the range 1 to $\ell - 1$, with ℓ the order of the cyclic group, so that public keys are transformed by simply multiplying by t.

The blinded key can then be used to index the descriptors while they are held by the HSDir. Clients can derive the blinded key from the `.onion` address and query for a descriptor by providing the blinded key. So, the blinded key serves as a private *index* from which the descriptor may be queried. This also implicitly means that the client is implicitly checking the connection between the identity public key from the `.onion` address and the blinded public key. For security it is important that only the actual owner of the `.onion` address can upload a descriptor to a given index. This is where the signing functionality of key-blinding is used. Onion services also upload a signature on the descriptor, which can be verified with the blinded key. When HSDirs verify this signature, they ensure that the descriptor is being uploaded by the actual owner of the identity public key — all without knowing what the `.onion` address is.

A malicious actor with a quantum computer could forge a signature with respect to a chosen blinded public key, and use this to upload false information about an introduction point. This would mean that queries to the onion service could be redirected to the adversary.

2.2 Key-Blinding Signature Scheme Definitions

Definition 1. *A key-blinding signature scheme* Δ *consists of four algorithms* $(\mathsf{KeyGen}, \mathsf{BlindPk}, \mathsf{Sign}, \mathsf{Verify})$ *where*

- $\mathsf{KeyGen}()$ *generates an identity key pair* (pk, sk).
- $\mathsf{BlindPk}(pk, \tau)$ *deterministically generates a blinded public key* pk_τ.
- $\mathsf{Sign}(m, sk, \tau)$ *may deterministically or probabilistically generate a signature* σ *for the message* m *using the identity secret key* sk *and epoch* τ.
- $\mathsf{Verify}(m, \sigma, pk_\tau)$ *accepts if the signature is valid under the message* m *and epoch* τ *used to generate* pk_τ*, otherwise it rejects.*

We require the usual correctness properties for signature schemes, but extended for key-blinding. That is, if (pk, sk) is a keypair generated from KeyGen, pk_τ is then derived from $\mathsf{BlindPk}$ with a given nonce τ, and $\sigma \leftarrow \mathsf{Sign}(m, sk, \tau)$, then with overwhelming probability $\mathsf{Verify}(m, \sigma, pk_\tau)$ will accept. Anyone without knowledge of the identity public key can verify using the pk_τ given in the descriptor, while someone with knowledge of the identity key can take the additional step of checking $pk_\tau = \mathsf{BlindPk}(pk, \tau)$. Note that we do not require that blinded keys can be blinded again.

Signatures with key-blinding must satisfy two security requirements. First, they must be *unlinkable*, which means that an adversary without knowledge of the identity public key who observes many public key blindings as well as signatures under those blindings cannot distinguish a fresh blinding of the public key from an entirely unrelated key. Second, the scheme must satisfy *unforgeability*. This property is largely the same for signature schemes with key-blinding as it is for typical signature schemes. However, rather than just devising an (m, σ) such that $\mathsf{Verify}(m, \sigma, pk)$ accepts, the adversary must be able to provide an (m, σ, τ) such that $\mathsf{Verify}(m, \sigma, pk_\tau)$ accepts where $pk_\tau \leftarrow \mathsf{BlindPk}(pk, \tau)$.

Earlier versions of both of these formulations appear in [24]. The security definitions that we present here are more general. The definitions in [24] were tied to the exact usage of key-blinding in Tor, and do not consider security in situations where the blinding process is decoupled from the signing process, so that multiple signatures can be issued under the same blinded public key.

Definition 2 (Unlinkability). *Let $\Delta = (\mathsf{KeyGen}, \mathsf{BlindPk}, \mathsf{Sign}, \mathsf{Verify})$ be a key-blinding signature scheme. Define $\mathbf{Exp}_\Delta^{\mathsf{UL-CMEA}}(\mathcal{A})$ as follows:*

- *Let $(pk, sk) \leftarrow \mathsf{KeyGen}()$ be freshly generated identity keys.*
- *$\mathcal{A}$ may query τ to a public key-blinding oracle to get $pk_\tau \leftarrow \mathsf{BlindPk}(pk, \tau)$.*
- *$\mathcal{A}$ may query (m, τ) to a signing oracle to receive $\sigma_{m,\tau} \leftarrow \mathsf{Sign}(m, sk, \tau)$ for any τ previously queried to the public key-blinding oracle.*
- *$\mathcal{A}$ makes a challenge query τ^* not previously queried to the public key-blinding oracle. A bit b is uniformly sampled, $pk_0 \leftarrow pk$, and a fresh pair of identity public keys $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}()$ is generated. $\mathcal{A}$ receives $pk_b^* \leftarrow \mathsf{BlindPk}(pk_b, \tau^*)$.*
- *$\mathcal{A}$ may additionally query the public key-blinding oracle or the signing oracle, except that if the queried $\tau = \tau^*$, the oracles use pk_b^*.*
- *$\mathcal{A}$ provides a bit b^* after expending t-bounded computational resources, q_B-bounded public key-blinding oracle queries, and q_S-bounded signing queries; and the game outputs 1 if $b^* = b$, otherwise it outputs 0.*

The $\mathsf{UL-CMEA}$ (unlinkability under chosen message and epoch attack) advantage is defined as $\mathbf{Adv}_\Delta^{\mathsf{UL-CMEA}}(\mathcal{A}) = \left|\Pr[\mathbf{Exp}_\Delta^{\mathsf{UL-CMEA}}(\mathcal{A}) = 1] - \frac{1}{2}\right|$.

Definition 3 (Unforgeability). *Let $\Delta = (\mathsf{KeyGen}, \mathsf{BlindPk}, \mathsf{Sign}, \mathsf{Verify})$ be a key-blinding signature scheme. Define $\mathbf{Exp}_\Delta^{\mathsf{EUF-CMEA}}(\mathcal{A})$ as follows:*

- *Let $(pk, sk) \leftarrow \mathsf{KeyGen}()$ be freshly generated identity keys.*

– $\mathcal{A}$ *may query* (m, τ) *to a signing oracle which generates* $pk_\tau \leftarrow \mathsf{BlindPk}(pk, \tau)$ *and* $\sigma_{m,\tau} \leftarrow \mathsf{Sign}(m, sk, \tau)$*, and sends* $(pk_\tau, \sigma_{m,\tau})$ *to* $\mathcal{A}$*.*
– $\mathcal{A}$ *submits* (m^*, σ^*, τ^*) *after expending* t*-bounded computational resources and* q_S*-bounded signing queries, and the game outputs* 1 *if* (m^*, τ^*) *was not previously queried and* $\mathsf{Verify}(m^*, \sigma^*, \mathsf{BlindPk}(pk, \tau^*)) = 1$*, otherwise it outputs* 0*.*

The $\mathsf{EUF-CMEA}$ *(existential unforgeability under chosen message and epoch attack) advantage is defined as* $\mathbf{Adv}_{\Delta}^{\mathsf{EUF-CMEA}}(\mathcal{A}) = \Pr[\mathbf{Exp}_{\Delta}^{\mathsf{EUF-CMEA}}(\mathcal{A}) = 1]$.

3 Unlinkability of Signature Schemes with Key-Blinding

We want to establish that an adversary who has access to a blinding oracle and a signing oracle still cannot distinguish a new blinding of the identity public key from the blinding of a fresh public key. We observed a common technique that could be used for showing unlinkability among the signature schemes we consider. To establish unlinkability, we devise a property we call *independent blinding*, which asks that the distribution of the output of the blinding function is independent from its input. This means that seeing any number of blindings of a public key leaks no information on the identity public key.

While our techniques provide a generic framework to establish unlinkability, they do not extend to showing unforgeability or provide a way to generically *construct* schemes with key-blinding out of Fiat–Shamir style singature schemes. This is because the mechanism by which blinding is accomplished changes depending on the scheme. As a result, there is no common framework for constructing a key-blinding scheme, and the proof of unforgeability similarly must take the blinding mechanism into account.

To guarantee that the signing oracle leaks no information about the identity public key, we require that the distribution of signatures is dependent only on the public key. This is best characterized by a property we call *signing with oracle reprogramming*, which states that if we have the ability to reprogram the random oracle used in the signature scheme, then we can create signatures indistinguishable from real ones for any message.

Many signature schemes show their security by first establishing just such a property. As an example, for signature schemes built from an identification protocol and the Fiat–Shamir heuristic, the zero-knowledge property is typically proven by establishing the ability to simulate transcripts given only the public key. When given control of the random oracle, we can sample transcripts and reprogram the random oracle to generate a signature.

To formalize this notion, we require a concept we call a *reprogrammed point extractor*. This is a simple function, efficiently computable and publicly known to all, which, given a signature σ, public key, and message, can extract the point on which the random oracle is reprogrammed to make the signature verify.

It is best to illustrate this with an example. Consider a generic form of the Probabilistic Signature Scheme [7] defined with respect to a trapdoor permutation T. To sign a message, sample a random salt r and compute $x = T^{-1}(H(pk \| m \| r))$.

The signature is $\sigma = (x, r)$. To verify a signature, simply check that $T(x) = H(pk\|m\|r)$. It is straightforward to show that signatures can be generated if the random oracle can be reprogrammed. On input of a message m, sample a random (x, r) and reprogram H so that $H(pk\|m\|r) = T(x)$. If T is a permutation, it is easy to see that (x, r) will have the same distribution as in a real signature, and the reprogramming cannot be detected as long as r is sufficiently long (so that the adversary is unlikely to have queried $pk\|m\|r$ beforehand). Let Ext denote our reprogrammed point extractor. For the example above, we have $\mathsf{Ext}((x, r), pk, m) = pk\|m\|r$.

Definition 4 (Signing with oracle reprogramming). *Let Σ be a signature scheme that relies on a random oracle H. We say that the signature scheme admits signing with oracle reprogramming if there exists a reprogrammed point extractor Ext and a forgery function Forge that takes in pk, m and returns (y, σ) such that $\Sigma.\mathsf{Verify}^{H:\mathsf{Ext}(\sigma,pk,m)\mapsto y}(pk, m, \sigma) \to$ 'accept', where $H : x \mapsto y$ denotes the random oracle reprogrammed such that $H(x) = y$.*

In order to use oracle reprogramming to sign a message, we need to consider the probability that an adversary is capable of noticing that the real signing algorithm wasn't used. This amounts to considering the joint distribution of the signature as well as the input and output of the hash function on the reprogrammed point.

Definition 5. *Let $\Sigma = (\mathsf{Sign}, \mathsf{Verify})$ be a signature scheme defined with respect to a random oracle H and a public key space $\mathcal{PK}$ that admits signing with oracle reprogramming via a point extractor Ext and a forgery function Forge.*

For a public key and message pk, m, we consider the adversary's ability to distinguish the distribution of $(y_{forged}, \sigma_{forged}) \leftarrow \mathsf{Forge}(pk, m)$ from the distribution of $(y_{real}, \sigma_{real})$ where $\sigma_{real} \leftarrow \mathsf{Sign}(pk, m)$ and $y_{real} = H(\mathsf{Ext}(\sigma_{real}, pk, m))$ (i.e., the output of the hash on the input that would be reprogrammed).

We denote $L1$ distance between these distributions as δ, that is

$$\delta = \sum_{\sigma,y} \left| \Pr[\sigma_{real} = \sigma, y_{real} = y] - \Pr[\sigma_{forged} = \sigma, y_{forged} = y] \right|.$$

As well, we need to consider the ability of an adversary to detect that reprogramming has occurred. This can be evaluated by considering the min-entropy of the point that is reprogrammed, to ensure that the probability that an adversary queries this point prior to reprogramming is low. Let h_{min} denote the min-entropy of $\mathsf{Ext}(\sigma, pk, m)$, where $(y, \sigma) \leftarrow \mathsf{Forge}(pk, m)$.

Note that in the above definition we are implicitly assuming that the statistical distance and the entropy are not dependent on m, pk, or H. For all of the schemes that we construct this is the case. Even if these values were dependent on pk, m, or H, the scheme could still be secure as long as they were sufficiently small on average. However to simplify the proof and notation, our definition only considers schemes where they do not depend on pk, m, or H.

We now consider the unlinkability experiment $\mathbf{Exp}_{\Delta}^{\mathsf{UL-CMEA}}$. We will show a reduction from an adversary who makes queries to the signing oracle to an adversary who makes none.

Lemma 1. *Let Δ be a key-blinding signature scheme which admits signing with oracle reprogramming with $L1$ distance δ and min-entropy of reprogrammed points h_{min}. Let $\mathcal{A}$ be an adversary making q_B queries to the blinding oracle, q_S queries to the signing oracle, and q_H queries to the random oracle. Using $\mathcal{A}$, we construct an adversary $\mathcal{A}^{q_S=0}$ that makes no signing queries (i.e., a key-only adversary) for which $\mathbf{Adv}_{\Delta}^{\mathsf{UL-CMEA}}(\mathcal{A}) \leq \mathbf{Adv}_{\Delta}^{\mathsf{UL-CMEA}}(\mathcal{A}^{q_S=0}) + q_H q_S 2^{-h_{min}} + q_S\delta$.*

Proof. To construct the adversary $\mathcal{A}^{q_S=0}$ while relying on the adversary $\mathcal{A}$ as a subroutine, we must show how to handle queries to the blinding oracle and the signing oracle. For queries to the blinding oracle, $\mathcal{A}^{q_S=0}$ can simply pass along these queries to the blinding oracle provided to them.

To handle the signing queries, we rely on signing with oracle reprogramming. Whenever a signing query is made with respect to a blinded public key pk_{τ_i}, we reprogram the random oracle in order to provide a signature. Thus we need to consider the adversary's ability to distinguish that the secret key is not being used to sign messages. To realize this, the adversary either needs to observe that the oracle has been reprogrammed, or notice a difference in the observed distribution of some part of the signature.

To distinguish that reprogramming has occurred during signing, the adversary must have queried the random oracle on the reprogrammed point previously. In total, q_S points will be reprogrammed. So the adversary makes q_H guesses, and then q_S points are chosen to be reprogrammed from a distribution with min entropy h_{min}, and we want to consider the probability of a match between the q_H and q_S points. We can upper-bound this by $q_H q_S 2^{-h_{min}}$.

Next we consider the output distribution of the programmed points. There are q_S reprogrammed points, and the statistical ($L1$) distance between the forged values and the real values is δ, so the adversary's advantage in distinguishing based on the distribution of reprogrammed values is at most $q_S\delta$. □

We now only need to consider the advantage of $\mathcal{A}^{q_S=0}$, an adversary who makes no queries to the signing oracle. So, we need only consider how the blinding oracle and random oracle provide information to the adversary.

To characterize the security of blinding, we want to insist that the distribution of the public key returned by $\mathsf{BlindPk}$ is independent of the identity public key input, so that no knowledge is gained. However care must be taken here, because the $\mathsf{BlindPk}$ algorithm is actually deterministic on the inputs pk and τ. So when we refer to the 'distribution' of $\mathsf{BlindPk}$ we need to be clear over what randomness.

In practice, the $\mathsf{BlindPk}$ function hashes the public key and the nonce τ to generate some randomness, and then uses that randomness to blind the public key. To separate out the process of hashing to generate randomness and using the randomness, we will define a new function $\mathsf{randBlind}(pk; r)$, which takes in a public key and some randomness, and blinds the public key. Then $\mathsf{BlindPk}$ is defined

by making $\mathsf{randBlind}$ deterministic through the random oracle H. Specifically, $\mathsf{BlindPk}(pk, \tau) = \mathsf{randBlind}(pk; H(pk\|\tau))$.

Definition 6 (Independent Blinding). *Let Δ be a key-blinding signature scheme and let n be a positive integer. Let $pk_0, pk_1, \ldots, pk_n$ be public keys generated from* KeyGen*. Sample uniform randomness $r_1, r_2, \ldots, r_n$. The* independent blinding advantage*, denoted $\mathbf{Adv}^{\mathsf{Ind-Blind}}_{\Delta,n}(\mathcal{A})$, is the advantage that an adversary has in distinguishing the following two distributions:*

1) $\mathsf{randBlind}(pk_0; r_1), \mathsf{randBlind}(pk_0; r_2), \ldots, \mathsf{randBlind}(pk_0; r_n)$
2) $\mathsf{randBlind}(pk_1; r_1), \mathsf{randBlind}(pk_2; r_2), \ldots, \mathsf{randBlind}(pk_n; r_n)$

This ensures that the adversary $\mathcal{A}^{q_S=0}$ may observe many blindings of the public key with respect to arbitrary nonces but what they see is close to a distribution independent of the identity public key.

Lemma 2. *Let Δ be a key-blinding signature scheme and let h_{pk} be the min-entropy of the public key returned from $\Delta.\mathsf{KeyGen}$. Let $\mathcal{A}^{q_S=0}$ be an $\mathsf{UL-CMEA}$ adversary that makes no queries to its signing oracle. Then there exists an algorithm $\mathcal{B}$ such that $\mathbf{Adv}^{\mathsf{UL-CMEA}}_{\Delta}(\mathcal{A}^{q_S=0}) \leq \mathbf{Adv}^{\mathsf{Ind-Blind}}_{\Delta,n}(\mathcal{B}) + q_H 2^{-h_{pk}}$, where n is the number of blinding queries $\mathcal{A}^{q_S=0}$ makes to its public key-blinding oracle and the runtime of $\mathcal{B}$ is approximately the same as the runtime of $\mathcal{A}$.*

Proof. We use a simple game-hopping proof to bound the adversary's success probability. Game G_0 proceeds according to $\mathbf{Exp}^{\mathsf{UL-CMEA}}$ with the adversary making no signing queries by assumption. In game G_1, when the adversary queries the blinding oracle with input τ, rather than responding with $\mathsf{BlindPk}(pk, \tau) = \mathsf{randBlind}(pk, H(pk\|\tau))$, we sample a uniformly random r and return $\mathsf{randBlind}(pk, r)$. Note that there is no difference between these games until an adversary queries $H(pk\|\tau)$ for some τ; we let bad be the event that the adversary makes such a query. Games G_0 and G_1 are identical-until-bad [6].

In game G_2 we modify the response to each blinding query from $\mathsf{randBlind}(pk, r)$ by sampling a fresh pk' each time from KeyGen and returning $\mathsf{randBlind}(pk', r)$. We can construct, from an adversary that distinguishes G_1 from G_2, a reduction $\mathcal{B}$ that distinguishes the two distributions in the independent blinding property: G_1 uses the first distribution in Definition 6, whereas G_2 uses the second. Thus G_2 can be distinguished from G_1 with advantage at most $\mathbf{Adv}^{\mathsf{Ind-Blind}}_{\Delta,n}(\mathcal{B})$.

We now consider the probability of event bad—i.e., the adversary querying $H(pk\|\tau)$—in G_2. Since none of the blindings actually use pk, the success probability is bounded by the adversary's ability to guess the public key. For this we use the min-entropy of the public key returned from key-generation. Over q_H queries, the probability that an adversary is able to guess the public key is bounded by $q_H 2^{-h_{pk}}$. By the fundamental lemma of game playing [6], this is the probability that an adversary is able to distinguish between game G_0 and G_1.

Finally, in game G_2 all blinded public keys are independent of the original key, so everything the adversary sees is independent of the challenge bit b, and thus the adversary's advantage in G_2 is 0, yielding the desired result. □

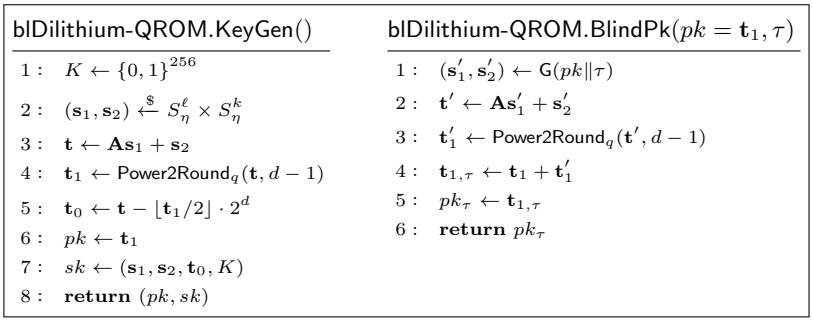


Fig. 2. Key generation and blinding algorithms for blDilithium-QROM.

One could go to the effort of computing or bounding the min-entropy h_{pk} of the public key returned from KeyGen for each scheme. It is convenient to observe that $2^{-h_{pk}} \leq \mathbf{Adv}_{\Delta}^{\mathsf{EUF-CMEA}}(\mathcal{A})$ for any adversary $\mathcal{A}$: otherwise, for a scheme where certain public keys have abnormally high change of being generated, an adversary could break unforgeability by repeatedly running KeyGen until the desired public key (and a corresponding secret key) is generated. Thus,

Corollary 1. *Let Δ and $\mathcal{A}^{q_S=0}$ be as in Lemma 2. Then there exist algorithms $\mathcal{B}_1, \mathcal{B}_2$ such that $\mathbf{Adv}_{\Delta}^{\mathsf{UL-CMEA}}(\mathcal{A}^{q_S=0}) \leq \mathbf{Adv}_{\Delta,n}^{\mathsf{Ind-Blind}}(\mathcal{B}_1) + q_H \mathbf{Adv}_{\Delta}^{\mathsf{EUF-CMEA}}(\mathcal{B}_2)$, where n is the number of blinding queries $\mathcal{A}^{q_S=0}$ makes to its public key-blinding oracle and the runtimes of $\mathcal{B}_1$ and $\mathcal{B}_2$ are approximately the same as that of $\mathcal{A}$.*

4 A Lattice-Based Key-Blinding Scheme

Dilithium [19] is a finalist in the NIST post-quantum signature standardization process and comes from a long line of lattice-based signature schemes. We present a key-blinded version of Dilithium-QROM [27] which modifies Dilithium to permit lossy key generation, hence allowing a reduction from the scheme to Module Learning with Errors (MLWE) assumption. Later, in Section 4.3, we discuss the challenges in blinding Dilithium itself.

Our construction, blDilithium-QROM utilizes the fact addition is homomorphic. As a result, the $\mathbf{A}$ matrix is a public matrix used by all parties in the network. In addition, both signing and verification use the public key when sampling the challenge c. Finally, the identity public key consists consists of an extra bit as this permits key-blinding.

4.1 blDilithium-QROM Description

We make use of functions defined in [27]. In addition, our notation mirrors that in [27]. A complete description is in Appendix A.

Signing is performed by using G to sample blinding secrets adding these to the identity public key secrets, and performing the operations in KeyGen. Then,

the procedure in [27] is followed except that the public key is added to the hash to produce the challenge c. Verification is also similar to that in [27] except that $c\mathbf{t}_{1,\tau}$ is multiplied by 2^{d-1}. The full scheme is supplied in Appendix A.

The parameters are identical to the parameters in [27], except that $d = 7$ and $\beta = 644$.

4.2 blDilithium-QROM Security

We now consider the security of blDilithium-QROM by addressing unforgeability and unlinkability as defined in Section 2. Since the proof of unforgeability closely follows the proof of unforgeability for Dilithium-QROM in [27], we give a summary here and provide the detailed proof in Appendix A. In addition, the proof of unlinkability can be found in Appendix A due to space constraints.

We first address unforgeability, in which we follow the framework set out in [27]. To begin, we create a version of blDilithium-QROM where the identity public key $pk = \mathbf{t}$, which we then use to construct an identification protocol ID whose Fiat-Shamir transform is equivalent to the scheme with the larger public key. We then follow the techniques in [27] to show that ID is non-abort honest verifier zero knowledge (naHVZK) and lossy, and establish bounds on its correctness and min-entropy. This allows us to use Theorem 3.1 of [27] to establish the following bound:

Theorem 1. *Let $\mathcal{A}$ be any adversary that makes at most q_{H} hash queries and q_S signing queries against the unforgeability of* blDilithium-QROM *with parameters as specified in Subsection 4.1. Then there exists an algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{k,\ell,\mathcal{U}}(\mathcal{B}) + 8(q_{\mathsf{H}}+1)\cdot 2^{-137} + 2^{-2899}$$

In general, the proof of naHVZK and bounds on correctness and min-entropy are identical to those in [27], except that the blinding factor is introduced and $\|2cs\|_\infty$ must be bounded by β. Lossiness differs in that the added blinding factor contributes to the bound on size of the solutions of a specific equation, hence raising the bound $\varepsilon_{\mathsf{ls}}$.

We now turn our attention to unlinkability and discuss independent blinding and signing with oracle reprogramming discussed in Section 3.

Theorem 2. *For any adversary $\mathcal{A}$ that makes q_S signing queries and q_{H} random oracle queries, there exists an algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{UL-CMEA}}_{\mathsf{blDilithium\text{-}QROM},t}(\mathcal{A}) \leq 2t\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\mathcal{U},\mathbf{A}}(\mathcal{B}_1) + q_{\mathsf{H}}\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{B}_2) + q_{\mathsf{H}}q_S 2^{-2899}$$

At a high level, the theorem follows from the fact a blinded public key $\mathbf{t}+\mathbf{t}'$ can be replaced by $\mathbf{t}+\tilde{\mathbf{t}}'$ where $\tilde{\mathbf{t}}'$ is uniformly sampled. Since we are working in R_q^k, then $\mathbf{t}+\tilde{\mathbf{t}}'$ is itself uniformly random. We make t hops away from independent blindings of a single public key using the replacement as above, then use another t hops to return to independent blindings of independent public keys.

We also know that blDilithium-QROM permits signing with oracle reprogramming as we can use the simulator from unforgeability to create a forgery function

with $\delta = 0$ and the min-entropy bound comes directly from the min-entropy bound in unforgeability.

As we make no changes to the set of parameters that contributes to MLWE hardness, the classical and quantum bit-security from [27] apply here. In particular, they argue that as there is no known attack that leverages the module structure of the assumption, it is common to directly apply LWE bit security directly to MLWE security. We make the further assumption that the SA-MLWE assumption is also as secure as the MLWE assumption.

4.3 Key-blinding Dilithium

We briefly describe a key-blinded version of Dilithium [19] but we provide no security analysis or guarantees.

As is with blDilithium-QROM, $\mathbf{A}$ is a public parameter of the network and thus ρ can be omitted from the scheme. In addition, Power2Round is modified to release one extra bit for $\mathbf{t}_1$ while keeping $\mathbf{t}_0$ the same. The appropriate changes to Sign and Verify are made in a similar fashion to the changes made from Dilithium-QROM to blDilithium-QROM.

Note that during signing, tr may be recomputed as it is dependent solely on the identity public key and not the blinded public key. One possibility could be to set $tr = \mathsf{CRH}(\mathbf{s}_{1,\tau}, \|\mathbf{s}_{2,\tau})$.

No parameters need to be changed to modify the correctness of the blinded scheme.

5 An Isogeny-Based Key-Blinding Scheme

In this section we briefly describe how to realize a key-blinding signature scheme from CSI-FiSh [14], which is an isogeny-based signature scheme that uses the structure of the CSIDH [15] group action. The 'group' here refers to class group $\mathrm{Cl}(\mathcal{O})$, with $\mathcal{O}$ being the endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E)$, the ring of endomorphisms from a curve E to itself defined over $\mathbb{F}_p$, which is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. A main contribution of the CSI-FiSh paper was to calculate the precise structure of this group, so that it can be described as a cyclic group of order N. This allows for two crucial operations with respect to the group action: group elements can now be sampled *uniformly* from the group, and group elements can now be given a *canonical representation* as a member of $\mathbb{Z}_N$, so that for example, when revealing to an adversary a group element $g = g_1 \cdot g_2$, we can be assured that no information about g_1 or g_2 is leaked by how g is represented.

For our purpose, we will describe the scheme as an abstract group action, and avoid notation that refers to how the group is actually constructed. For complete details about the group action we refer to the CSI-FiSh paper [14].

We briefly recall the details of a group action. We have a group G and a set $\mathcal{E}$ along with an operation $\star : G \times \mathcal{E} \to E$. The operation $\star$ satisfies the property that if $id \in G$ is the identity group element, then $id \star E = E$ for all $E \in \mathcal{E}$.

Furthermore, for $g_1, g_2 \in G$, it must be the case that $g_1 \star (g_2 \star E) = (g_1 \cdot g_2) \star E$. In fact, the group action described in [14] is both *free* and *transitive*, meaning that for $E_1, E_2 \in \mathcal{E}$ there is one and only one $g \in G$ such that $g \star E_1 = E_2$. Furthermore, the group G in our case is cyclic, and we will denote the order N.

In CSI-Fish, signatures correspond to a zero-knowledge proof of knowledge of a secret group element g_{sk} such that $g_{sk} \star E_0 = E_{pk}$, with E_{pk} being the public key and E_0 being a system parameter. Proving knowledge of such a g_{sk} is done via a simple sigma protocol. The commitment is created by uniformly sampling $g \xleftarrow{\$} G$ and computing $E_{com} = g \star E_{pk}$ as the commitment. The verifier then selects a bit $b \xleftarrow{\$} \{0, 1\}$ as the challenge, and the prover responds by sending g if $b = 0$, and $g \cdot g_{sk}$ if $b = 1$.

The verifier then checks: if $b = 0$ that $g \star E_{pk} = E_{com}$; and if $b = 1$ that $(g \cdot g_{sk}) \star E_0 = E_{com}$. Soundness follows from the fact that, from two responses g and $g \cdot g_{sk}$, the secret key g_{sk} can quickly be recovered. Honest-verifier zero-knowledge can be shown by simulating transcripts in a straightforward way (here we rely on the fact that group elements have a canonical representation).

The basic idea of how key-blinding functionality can be added to the scheme is already apparent. From a value τ, a group element g_τ can be generated, and the public key E_{pk} is blinded to $E_\tau = g_\tau \star E_{pk}$. Anyone who knows the public key and τ can perform this operation, but to sign a message, one must know g_τ and g_{sk} so the scheme is still unforgeable. Furthermore, because the group action is transitive, the action of g_τ entirely hides E_{pk}. Observing many blindings still leaks no information about E_{pk}, ensuring that the scheme is unlinkable.

Of course, the soundness of this zero-knowledge scheme is only $1/2$, and would have to be repeated many times in order for the signature scheme to be existentially unforgeable. The authors of CSI-FiSh employed many clever techniques in order to improve on the efficiency of the scheme over just repeating the signature scheme 128 times. Most notably, the public keys of CSI-FiSh consist of many curves $E_{pk,1}, E_{pk,2}, \ldots, E_{pk,L}$, generated by computing $g_{sk,1} \star E_0$, $g_{sk,2} \star E_0$, etc. Then rather than choosing a single bit for the challenge, an index from 0 to L can be chosen. This increases the soundness significantly, and so the protocol can be repeated fewer times to achieve the same level of security, allowing for a trade-off between the signature size and the public key size. To blind, we can similarly sample independent blinding factors $g_{\tau,1}, g_{\tau,2}, \ldots$ and apply each of them to each part of the public key.

A further advantage that CSI-FiSh optionally takes is to then 'Merkleize' the public key. Rather than including each of $E_{pk,1}, \ldots, E_{pk,L}$, key generation commits to these public keys by constructing a Merkle tree with each curve as a leaf node. When signing a message, each $E_{pk,i}$ that gets used, as well as the Merkle path that proves the commitment, is provided. This causes the public key to be only 32 Bytes, at the expense of increasing the size of signatures and making signing and verification slightly slower. Unfortunately, it is not possible to use this technique for a blinded version. The raw $E_{pk,i}$ values must be available in order to construct the blinded version of the public key, and so 'Merkleization' is impossible.

blCSI-FiSh.KeyGen()	blCSI-FiSh.BlindPk($(E_{pk,i})_{i\in[L]}, \tau$)
1 : **for** $i \in [L]$ **do**	1 : $(g_{\tau,i})_{i\in[L]} \leftarrow KDF((E_{pk,i})_{i\in[L]} \| \tau)$
2 : $g_{sk,i} \xleftarrow{\$} \mathbb{Z}_N$	2 : **for** $i \in [L]$ **do**
3 : $E_{pk,i} \leftarrow g_{sk,i} \star E_0$	3 : $E_{\tau,i} \leftarrow g_{\tau,i} \star E_{pk,i}$
4 : **endfor**	4 : **endfor**
5 : **return** $(pk, sk) = ((E_{pk,i})_{i\in[L]}, (g_{sk,i})_{i\in[L]})$	5 : **return** $pk_\tau = (E_{\tau,i})_{i\in[L]}$

In Appendix D we describe signing and verification, which are essentially unchanged from in CSI-FiSh. There we also prove the unlinkability of the scheme, and discuss the proof of unforgeability of the scheme.

6 A Number-theoretic Key-Blinding Scheme

LegRoast and PorcRoast are new adaptations of Picnic that use the Legendre Symbol as a symmetric PRF [12,17]. In this section we show how the mathematical structure of LegRoast enables a more efficient key-blinding signature scheme.

Recall that the Legendre symbol modulo a prime p, denoted $(\frac{a}{p})$, is defined as 0 if $a \equiv 0 \pmod p$, 1 if a is a quadratic residue modulo p, and -1 if it is not. To use the Legendre symbol as a 1-bit keyed PRF with input X and key K, we can define a function that returns values in $\{0, 1\}$. For an odd prime p, define $\mathcal{L}_K(X)$ to return 0 if $K + X$ is a quadratic residue or $0 \pmod p$, and 1 otherwise. This concept can be generalized to consider the ℓ-th power residue, instead of just quadratic residues. This allows for a keyed PRF with $\log \ell$ bits of output to be defined as

$$\mathcal{L}^\ell_K(X) = \begin{cases} i, & \text{if } (X+K)/g^i \equiv h^\ell \pmod p \text{ for some } h \in \mathbb{F}_p^\times \\ 0, & \text{if } K + X \equiv 0 \pmod p. \end{cases}$$

A key property of this PRF is that it is a group homomorphism from $\mathbb{F}_p^\times$ to $\mathbb{Z}_\ell$. This is helpful for proving statements in zero-knowledge about preimages of the PRF. To prove knowledge of a K such that $\mathcal{L}^\ell_K(X) = s$, one can sample a random value $r \in \mathbb{F}_p^\times$ and send $(K + X) \cdot r$ and $\mathcal{L}^\ell_0(r)$. The prover then only needs to prove that the multiplication of $(K + X) \cdot r$ was computed correctly for the verifier to calculate s and be convinced of knowledge of K.[1] Since the equation being proven consists of a single multiplication gate, the resulting proof can be comparatively short.

LegRoast and PorcRoast [12] expand this idea into a signature scheme that uses the Fiat–Shamir heuristic. Public keys consist of the output of L computations of the Legendre PRF, with inputs $\mathcal{I} = i_1, \ldots, i_L$, which can be public

[1] The verifier must also be convinced that the prover did not lie about the value of $\mathcal{L}^\ell_0(r)$. This is accomplished by having the prover commit to this value before the challenge X is issued, so that the prover cannot choose the output of the PRF in a way to help them.

Algorithm 1 blLegRoast.BlindPk

Input: identity public key $pk = (w_1, w_2, \ldots, w_L)$, epoch τ
Output: blinded public key $pk_\tau = (w_{1,\tau}, w_{2,\tau}, \ldots, w_{L,\tau})$
1: $T \leftarrow KDF(pk||\tau)$
2: $v_1 \leftarrow \mathcal{L}^\ell_T(j_1), \ldots, v_L \leftarrow \mathcal{L}^\ell_T(j_L)$
3: $w_{1,\tau} \leftarrow w_1 + v_1 \pmod{\ell}, \ldots, w_{L,\tau} \leftarrow w_L + v_L \pmod{\ell}$
4: **return** pk_τ

parameters. We define the function F, which is parameterized by ℓ and $\mathcal{I}$, as taking in the secret key K and returning $\mathcal{L}^\ell_K(i_m)$ for $m \in [L]$. Hence, key generation consists of sampling a random secret key $K \in \mathbb{F}_p^\times$ and computing the public key $F^\ell_{\mathcal{I}}(K) = \big(\mathcal{L}^\ell_K(i_1), \ldots, \mathcal{L}^\ell_K(i_L)\big)$.

The same homomorphic property that makes the Legendre symbol an attractive option for zero knowledge proofs is also what allows for a blinding mechanism. Hashing the nonce and public key to a value $T \in \mathbb{F}_p^\times$, we can calculate L computations of the Legendre PRF with separate inputs $j_1, \ldots, j_L$. The public key blinded under the value T becomes $\big(\mathcal{L}^\ell_K(i_1) + \mathcal{L}^\ell_T(j_1), \ldots, \mathcal{L}^\ell_K(i_L) + \mathcal{L}^\ell_T(j_L)\big)$, where addition is performed modulo ℓ. Due to the homomorphic property of $\mathcal{L}$, this can also be written as $\big(\mathcal{L}^\ell_0((K + i_m) \cdot (T + j_m))\big)_{m \in [L]}$.

As mentioned, LegRoast works by presenting parts of the public key multiplied by random values $r^{(j)} \in \mathbb{F}_p^\times$, the results of which are denoted by $o^{(j)}$. Then the signer proves knowledge of K by presenting a zero knowledge proof that a random linear combination of B $(K + I^{(j)}) \cdot r^{(j)} - o^{(j)}$ terms is equal to 0; here the $I^{(j)}$ values are a random re-indexing of the $i^{(j)}$ values in the public key. We call such a linear combination the error term, which should be equal to zero. Once the coefficients $\{\lambda^{(j)}\}$ of the linear combination are defined, the error term is $E = \sum_{j=1}^B \lambda^{(j)} \left((K + I^{(j)}) \cdot r^{(j)} - o^j\right) = K \cdot \left(\sum_{j=1}^B \lambda^{(j)} r^{(j)}\right) + \sum_{j=1}^B \lambda^{(j)}(I^{(j)} r^{(j)} - o^{(j)})$. Since only the K and $r^{(j)}$ values are secret, the only time we have a secret value multiplied by a secret value is in the $K \cdot \sum \lambda^{(j)} r^{(j)}$ term, so this can be verified to be 0 with only one multiplication gate.

If we are using a blinded public key, then the corresponding error term is the summation of $\lambda^{(j)}((K+I^{(j)})(T+J^{(j)})r^{(j)} - o^{(j)})$ terms. Through rearranging in a similar way to that of LegRoast, we get an error term that has three multiplication gates as opposed to one. Due to the nature of the zero-knowledge proof system used, the complete description of signing and verifying is quite large, and so we move it to Appendix E. We focus on a description of the blinding process.

To complete the security assessment for blLegRoast, we still need to establish (i) the independent blinding property, (ii) the signing with oracle reprogramming property, and (iii) the existential unforgeability of the scheme. As the scheme uses the Fiat–Shamir heuristic, signing with reprogramming is possible by choosing the output of the hash function in advance and constructing the signature accordingly. The existential unforgeability of the scheme follows from how finding a K and T that satisfy the relations informed by the public key is still hard. As these proofs

require careful details of the scheme itself, we defer them to Appendix E, where the complete description of the scheme can be found.

7 Implementation Details

We implemented the blDilithium-QROM, the blCSI-FiSh, and blLegRoast schemes; code for each is available at `http://github.com/tedeaton/pq-key-blinding`. The code for blCSI-FiSh and blLegRoast is forked from the CSI-FiSh and LegRoast code respectively [11, 13] and is written primarily in C. The code for blDilithium-QROM is written in Sage. Results can be seen in Figure 1 in Section 1. Our performance metrics indicate that the increase over the unblinded version of schemes is quite reasonable.

blDilithium-QROM. For blDilithium-QROM, key generation and verification are in fact *faster* since a fixed parameter $\mathbf{A}$ is used for all users and can be pregenerated, rather than being pseudorandomly generated each time. The signing procedure of blDilithium-QROM is three times slower than that of Dilithium-QROM. We caution that, since our blDilithium-QROM implementation is written in Sage, the implementation is non-optimized and results not be used an absolute measure of performance, but can still give insight when *compared* to a similar Sage implementation of non-blinded Dilithium-QROM.

blLegRoast. Blinded LegRoast's performance is compelling both in absolute terms (under 1 ms for key generation and blinding, under 20 ms for signing and verifying) and comparative terms (no worse than 1.5× slower than unblinded LegRoast).

blPicnic. We leave an implementation of blPicnic as future work. New advancements to the zero knowledge protocol that Picnic uses are still being made [4], so the performance of the scheme, and any blinded version, will change. We can summarize what we expect to see in a blPicnic implementation however. Public keys should be maintained at a straightforward 32 bytes, which is very attractive. We do not have exact calculations for the signature size, but the circuit being used is twice as large (for two encryptions), so we would expect the size to be roughly twice as large. In practice it may not be quite twice as large, however, as some of the values sent are independent of the length of the circuit.

blCSI-FiSh. Our blCSI-FiSh implementation achieves sizes and performance effectively matching that of CSI-FiSh-unMerkleized. The CSI-FiSh and blCSI-FiSh implementations use the CSIDH-512 parameter set. This parameter set aims to achieve NIST level 1 security (comparable to the security of AES-128 against a quantum adversary), though whether it achieves this level of security has been a matter of contention [30]. Unfortunately, increasing the parameters in CSI-FiSh is a matter of great difficulty. It is essential to CSI-FiSh that the structure of the class group be known. Calculating the order N of the group was a subexponential computation that took the CSIDH authors 52 core years. If the parameters are increased, then a new computation must happen, which will almost certainly be infeasible. Quantum computers could calculate the structure of the class group much more easily, so by the time CSI-FiSh is needed, there may also be the ability to use it by computing the class group number.

Tor integration. Recall that Tor uses identity public keys as the URL for `.onion` addresses. This means that, unless the onion service lookup process changes, users directly interact with an onion service's public key, whether by clicking on it as a link or copying and pasting it into a browser window. This motivates keeping public keys as small as possible. For this purpose, blPicnic and blLegRoast are the most attractive of the schemes considered. In the context of Tor, the process for connecting to an onion service is quite lengthy (several seconds, usually), so there may be less sensitivity to increased computation time.

8 Conclusion

We have considered the problem of building post-quantum key blinding schemes. We have shown that the unlinkability property can be reduced to two properties that are often relatively easy to establish: that blinding properly re-randomizes the public key (independent blinding) and that the distribution of signatures is only dependent on the public key (signing with oracle reprogramming). We have shown four different ways that post-quantum key blinding can be achieved: with supersingular isogenies via CSI-FiSh, lattices via Dilithium-QROM, with only symmetric primitives via Picnic, and by a number theoretic construction via LegRoast. We implemented blDilithium-QROM, blCSI-FiSh, and blLegRoast, and saw small performance impact compared to the unblinded versions.

Each of these four schemes is built out of the Fiat–Shamir paradigm. We did not consider any schemes built out of other ways to build signature schemes, such as hash-based signatures like SPHINCS+ [9], or the hash-and-sign paradigm like Rainbow [18] or Falcon [32].

It is difficult to envision a hash-based key blinding scheme. As public keys are the root of a Merkle tree, the only simple operation to blind a public key would be to hash it again. This could satisfy independent blinding, but not signing with oracle reprogramming: hash-based signatures work by providing paths up to the root, so the identity public key would be revealed on that path.

Hash-and-sign algorithms appear to have the opposite problem. A blinded version would almost certainly satisfy the signing with oracle programming property. If the trapdoored function is F, then by choosing a point x in the domain of F and programming the hash function so that $H(msg) = F(x)$, we obtain a signature; this is how hash-and-sign signature schemes often prove security. But it is not clear how to justify the independent blinding property. The most simple blinding mechanism would be to compose the trapdoor function F with another mapping G based on the blinding factor. This requires the range of F to match the domain of G, which makes it an interesting problem to be used with a hash-and-sign scheme. As well, to ensure the independent blinding property, we need that $F \circ G$ cannot be decomposed into the two mappings, which is a more novel security assumption. Because RSA is a trapdoor *permutation*, the structure of its mapping may allow for key-blinding, but it is not clear if any post-quantum primitive immediately does.

For these reasons, signature schemes that follow the Fiat–Shamir paradigm appear to admit key blinding much more readily. While homomorphic properties over the key space are certainly useful for key blinding (as in Dilithium and CSI-FiSh), they are not actually necessary, as the Picnic construction shows.

Acknowledgements. E.E. was supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Alexander Graham Bell Canada Graduate Scholarship. D.S. was supported by NSERC Discovery grant RGPIN-2016-05146 and a Discovery Accelerator Supplement.

References

1. Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly secure signatures from lossy identification schemes. *J. Cryptol.*, 29(3):597–631, 2016.
2. Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In *ACM Conference on Computer and Communications Security (CCS) 2020*, pages 1017–1031, 2020.
3. Paulo S. L. M. Barreto, Jefferson E. Ricardini, Marcos A. Simplício Jr., and Harsh Kupwade Patil. qSCMS: Post-quantum certificate provisioning process for V2X. Cryptology ePrint Archive, Report 2018/1247, 2018.
4. 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*, volume 12710 of *LNCS*, pages 266–297, 2021.
5. 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*, volume 12110 of *LNCS*, pages 495–526, 2020.
6. Mihir Bellare and Philip Rogaway. Code-based game-playing proofs and the security of triple encryption. In *Eurocrypt 2006*, volume 4004 of *LNCS*, pages 40–426, 2006.
7. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *Eurocrypt '96*, volume 1070 of *LNCS*, pages 399–416, 1996.
8. Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *Cryptographic Hardware and Embedded Systems (CHES) 2011*, volume 6917 of *LNCS*, pages 124–142, 2011.
9. Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In *ACM Conference on Computer and Communications Security (CCS) 2019*, pages 2129–2146, 2019.
10. Ward Beullens, Tim Beyne, Aleksei Udovenko, and Giuseppe Vitto. Cryptanalysis of the Legendre PRF and generalizations. *IACR Trans. Symmetric Cryptol.*, 2020(1):313–330, 2020.
11. Ward Beullens and Cyprien Delpech de Saint Guilhem. Legroast. GitHub Repository. `https://github.com/WardBeullens/LegRoast`, 2020. Accessed May 2021.
12. Ward Beullens and Cyprien Delpech de Saint Guilhem. LegRoast: Efficient post-quantum signatures from the Legendre PRF. In *11th International Conference on Post-Quantum Cryptography*, volume 12100 of *LNCS*, pages 130–150, 2020.

13. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh. GitHub Repository. `https://github.com/KULeuven-COSIC/CSI-FiSh`, 2019. Accessed May 2021.
14. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Asiacrypt 2019*, volume 11921 of *LNCS*, pages 227–247, 2019.
15. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *Asiacrypt 2018*, volume 11274 of *LNCS*, pages 395–427, 2018.
16. 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 *ACM Conference on Computer and Communications Security (CCS) 2017*, pages 1825–1842, 2017.
17. Ivan Damgård. On the randomness of legendre and jacobi sequences. In *Crypto '88*, volume 403 of *LNCS*, pages 163–172, 1988.
18. Jintai Ding, Ming-Shing Chen, Albrecht Petzoldt, Dieter Schmidt, and Bo-Yin Yang. Rainbow, 2019. Available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.
19. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.
20. Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for boolean circuits. In *USENIX Security 2016*, pages 1069–1083, 2016.
21. Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. In *ACM Conference on Computer and Communications Security (CCS) 2016*, pages 430–443, 2016.
22. Gus Gutoski and Douglas Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In *Financial Cryptography and Data Security (FC) 2015*, volume 8975 of *LNCS*, pages 497–504, 2015.
23. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In *Crypto 2020*, volume 12171 of *LNCS*, pages 500–529, 2020.
24. Nicholas Hopper. Proving security of Tor's hidden service identity blinding protocol. `https://www-users.cs.umn.edu/~hoppernj/basic-proof.pdf`, 2013.
25. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *ACM Symposium on Theory of Computing (STOC) 2007*, pages 21–30, 2007.
26. Dmitry Khovratovich. Key recovery attacks on the Legendre PRFs within the birthday bound. Cryptology ePrint Archive, Report 2019/862, 2019.
27. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat–Shamir signatures in the quantum random-oracle model. In *Eurocrypt 2018*, volume 10822 of *LNCS*, pages 552–586, 2018.
28. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes, and Cryptography*, 75(3):565–599, 2015.
29. Zhen Liu, Khoa Nguyen, Guomin Yang, Huaxiong Wang, and Duncan S. Wong. A lattice-based linkable ring signature supporting stealth addresses. In *Computer Security – ESORICS 2019*, pages 726–746. Springer International Publishing, 2019.
30. Chris Peikert. He gives c-sieves on the CSIDH. In *Eurocrypt 2020*, volume 12106 of *LNCS*, pages 463–492, 2020.

31. Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A practical multivariate blind signature scheme. In *Financial Cryptography and Data Security (FC) 2017*, volume 10322 of *LNCS*, pages 437–454, 2017.
32. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon, 2019. Available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.
33. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *ACM Symposium on Theory of Computing (STOC) 2005*, pages 84–93, 2005.
34. The Tor Project, Inc. Tor Metrics. `https://metrics.torproject.org/`, 2020. Accessed May 2020.
35. The Tor Project, Inc. Tor Rendezvous Specification - Version 3. `https://gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt`, 2020.

A blDilithium-QROM Security Proof

A.1 blDilithium-QROM Preliminaries

Throughout this section, vectors are written in lowercase bold-face and matrices are written in uppercase bold-face. Let q be prime, then $\mathbb{Z}_q$ denotes the integers modulo q, and R and R_q denote the rings $\mathbb{Z}[x]/\langle x^n+1\rangle$ and $\mathbb{Z}_q[x]/\langle x^n+1\rangle$ respectively.

For some $w \in \mathbb{Z}_q$, $\|w\|_\infty$ denotes $\left|w'\right|$ where $w' \in \mathbb{Z}$ such that $w' \equiv w \pmod q$ and $-\frac{q-1}{2} \leq w' \leq \frac{w-1}{2}$. This norm can be extended as follows. For some $w = w_{n-1}x^{n-1} + \cdots + w_1 x + w_0 \in R$, $\|w\|_\infty = \max\left(\|w_1\|_\infty, \ldots, \|w_{n-1}\|_\infty\right)$, and for some $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$, $\|\mathbf{w}\|_\infty = \max\left(\|w_1\|_\infty, \ldots, \|w_k\|_\infty\right)$. We let S_η denote the set of $w \in R$ such that $\|w\|_\infty \leq \eta$. In addition, we also define the L2 norm for $w \in R$ as $\|w\| = \sqrt{\|w_0\|_\infty^2 + \cdots + \|w_{n-1}\|_\infty^2}$.

ChSet denotes the challenge space. In the context of identification protocols, ChSet is the set of possible challenges a verifier can submit to the prover. In the context of signature schemes arrived at via the Fiat-Shamir transform, ChSet is the output domain of the hash function H used to digest the message.

G is also a hash function that takes in elements of $R_q^k \times R_q^k \times \{0,1\}^*$ and returns elements of $S_\eta^\ell \times S_\eta^k$.

We make use of several supporting algorithms with full descriptions in [27] which extract or compute on higher and lower order bits of elements in $\mathbb{Z}_q$. These algorithms are extended to elements of R_q and R_q^k by coefficient-wise and element-wise application. We give a general description of the supporting algorithms:

- $\mathsf{Power2Round}_q(r, d)$ extracts the higher $(\log(r) - d)$ order bits of r.
- $\mathsf{HighBits}_q(r, \alpha)$ extracts the higher $(\approx \log(r) - \log\alpha)$ order bits of r.
- $\mathsf{LowBits}_q(r, \alpha)$ extracts the lower $(\approx \log\alpha)$ order bits of r.
- $\mathsf{MakeHint}_q(z, r, \alpha)$ constructs a hint to allow the computation of the higher order bits of $r + z$ without the need to store z.
- $\mathsf{UseHint}_q(h, r, \alpha)$ uses the hint to compute the higher order bits of $r + z$.

The interactions of these algorithms are outlined by Lemmas 4.1 and 4.2 of [27], which we make use of in the proof of unforgeability.

We now discuss the MLWE assumption which was introduced in [28] as a generalization of the LWE assumption introduced in [33]. We leverage the decision version of the assumption, which posits that the following problem is hard given appropriate parameter selection.

Definition 7. *The decisional* $\mathsf{MLWE}_{m,k,\chi}$ *problem over the ring* R_q *is to distinguish the pair* $(\mathbf{A}, \mathbf{t})$ *for* $\mathbf{A} \xleftarrow{\$} R_q^{m\times k}, \mathbf{t} \xleftarrow{\$} R_q^m$ *from the pair* $(\mathbf{A}, \mathbf{As_1} + \mathbf{s_2})$ *where* $\mathbf{A} \xleftarrow{\$} R_q^{m\times k}, \mathbf{s}_1 \xleftarrow{\$} \chi(R_q^k), \mathbf{s}_2 \xleftarrow{\$} \chi(R_q^m)$. *The* $\mathsf{MLWE}_{m,k,\chi}$ *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{MLWE}}_{m,k,\chi}(\mathcal{A}) = \left|\Pr[\mathcal{A} \textit{ wins decisional } \mathsf{MLWE}_{m,k,\chi} \textit{ over } R_q] - \frac{1}{2}\right|$$

We introduce a modified version of the MLWE assumption in which the $\mathbf{A}$ matrix is a parameter of decisional MLWE problem. This modified assumption is required because each signer in the anonymity network shares the same $\mathbf{A}$ matrix. We refer to this assumption as the MLWE assumption with static $\mathbf{A}$, which assumes that the following problem is hard given appropriate parameter selection.

Definition 8. *The decisional* $\mathsf{SA\text{-}MLWE}_{m,k,\chi,\mathbf{A}}$ *problem over the ring* R_q *is to distinguish* $\mathbf{t}$ *for* $\mathbf{t} \xleftarrow{\$} R_q^m$ *from* $\mathbf{As}_1 + \mathbf{s_2}$ *where* $\mathbf{s}_1 \xleftarrow{\$} \chi(R_q^k), \mathbf{s}_2 \xleftarrow{\$} \chi(R_q^m)$. *The* $\mathsf{SA\text{-}MLWE}_{m,k,\chi}$ *advantage is defined as*

$$\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\chi,\mathbf{A}}(\mathcal{A}) = \left|\Pr[\mathcal{A} \textit{ wins decisional } \mathsf{SA\text{-}MLWE}_{m,k,\chi,\mathbf{A}} \textit{ over } R_q] - \frac{1}{2}\right|$$

A.2 blDilithium-QROM Specification

We now write out a version of blDilithium-QROM in which both $\mathbf{t}_1$ and $\mathbf{t}_0$ are published in the identity public key. This step is necessary for both the proofs of unforgeability and unlinkability. Observe that signatures generated via the procedure in Figure 3 are identical to those generated in Figure 5 given the same inputs and randomness, hence the scheme in Figure 3 must be at least as secure as the scheme in Figure 5 as it simply releases less information in the identity public key.

A.3 blDilithium-QROM Unforgeability

We begin by proving the unforgeability of blDilithium-QROM in the context of key-blinded signature schemes (see Definition 3), which we achieve by emulating the proof found in [27] while introducing the blinding procedure into relevant algorithms. At a high level, we begin by defining an identification protocol blDilithium-QROM-ID, addressing four key properties (naHVZK, correctness, lossiness, and min entropy), and applying the Fiat-Shamir transform to arrive at a signature scheme equivalent to blDilithium-QROM. This allows us to leverage Theorem 3.1 of [27] to bound $\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{A})$.

Non Abort Honest Verifier Zero-Knowledge We begin by showing that blDilithium-QROM-ID is naHVZK with $\varepsilon_{\mathsf{zk}} = 0$ as defined in Definition 2.5 of [27]. This entails showing that transcripts of honest interactions of blDilithium-QROM-ID are statistically indistinguishable from the output of some transcript simulator that only has access to the public key.

Lemma 3. *If* $\max_{s \in S_\eta, c \in \mathsf{ChSet}} \|2cs\|_\infty \leq \beta$ *then* blDilithium-QROM-ID *is* naHVZK *with* $\varepsilon_{\mathsf{zk}} = 0$.

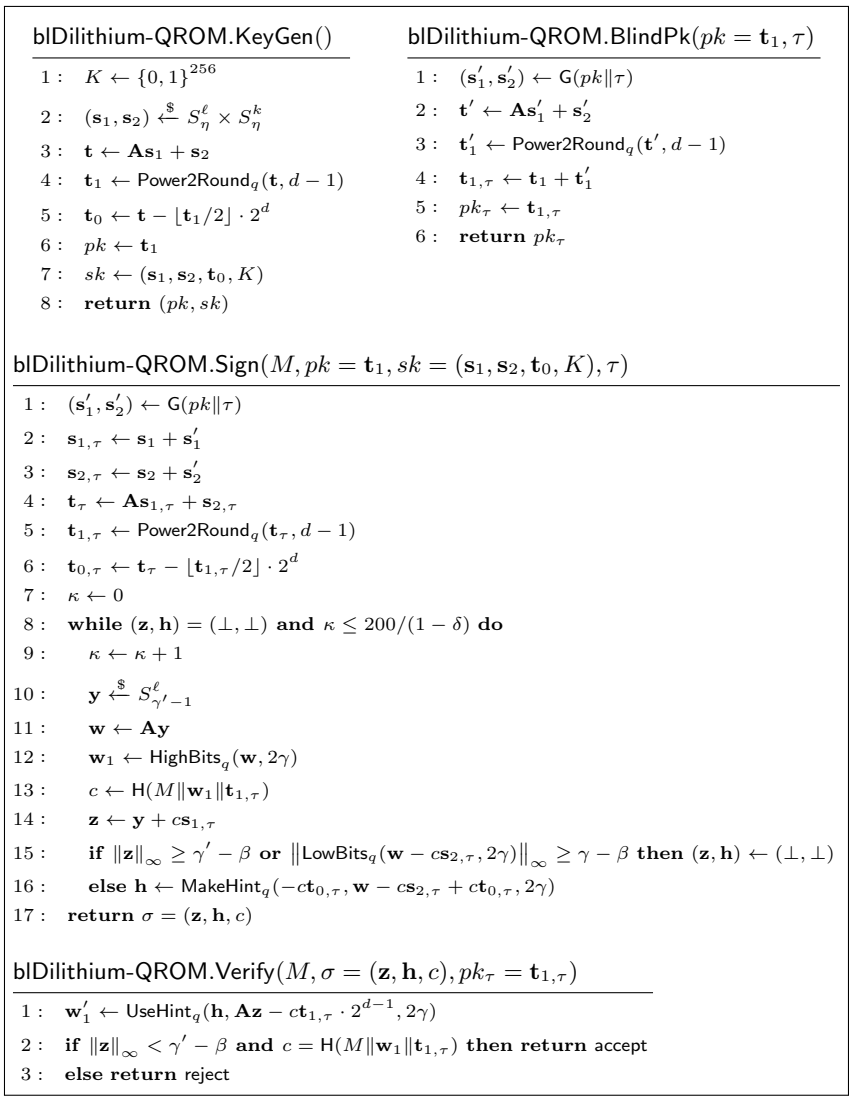


Fig. 3. Key generation, blinding, signing, and verification algorithms for blinded Dilithium-QROM (blDilithium-QROM) signature scheme.

	blDilithium-QROM recommended	Dilithium-QROM recommended
q	$2^{45} - 21283$	$2^{45} - 21283$
n	512	512
(k, ℓ)	(4,4)	(4,4)
d	7	15
$\mathsf{ChSet} = \{c \in R \mid \|c\|_\infty = 1 \wedge \|c\| = \dots\}$	$\sqrt{46}$	$\sqrt{46}$
γ	905679	905679
γ'	905679	905679
η	7	7
β	644	322
BKZ block-size to break LWE	480	
Best known classical bit-cost	136	
Best known quantum bit-cost	127	

Fig. 4. Parameters for the blDilithium-QROM scheme. Dilithium-QROM parameters are included for comparison

Proof. Suppose $\mathbf{s}_1, \mathbf{s}_2$ come from a valid identity keypair i.e. $\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{t}$. For a given $\mathbf{z} \in S^\ell_{\gamma'-\beta-1}$ and $c \in \mathsf{ChSet}$, the probability $\mathbf{z}$ was generated in Trans is equal to

$$\Pr[\mathbf{y} \xleftarrow{\$} S^\ell_{\gamma'-1} \mid \mathbf{y} + c(\mathbf{s}_1 + \mathbf{s}'_1) = \mathbf{z}] = \Pr[\mathbf{y} \xleftarrow{\$} S^\ell_{\gamma'-1} \mid \mathbf{y} = \mathbf{z} - c(\mathbf{s}_1 + \mathbf{s}'_1)]$$

Since $\left\|c(\mathbf{s}_1 + \mathbf{s}_2)\right\|_\infty \leq \beta$, then $\mathbf{z} - c(\mathbf{s}_1 + \mathbf{s}'_1) \in S^\ell_{\gamma'-1}$ thus

$$\Pr[\mathbf{y} \xleftarrow{\$} S^\ell_{\gamma'-1} \mid \mathbf{y} = \mathbf{z} - c(\mathbf{s}_1 + \mathbf{s}_2)] = \frac{1}{\left|S^\ell_{\gamma'-1}\right|}$$

Hence, every $\mathbf{z} \in S^\ell_{\gamma'-\beta-1}$ has an equal probability of being generated. In addition, it follows that the probability of not producing a $\mathbf{z} \in S^\ell_{\gamma'-\beta-1}$ in Trans is $1 - \left|S^\ell_{\gamma'-\beta-1}\right| / \left|S^\ell_{\gamma'-1}\right|$, so the distribution of $(c, \mathbf{z})$ is identical between Trans and Sim.

Finally, observe that

$$\mathbf{w} - c(\mathbf{s}_2 + \mathbf{s}'_2) = \mathbf{A}\mathbf{y} - c(\mathbf{s}_2 + \mathbf{s}'_2) = \mathbf{A}(\mathbf{z} - c(\mathbf{s}_1 + \mathbf{s}'_1)) - (\mathbf{s}_2 + \mathbf{s}'_2) = \mathbf{A}\mathbf{z} - c(\mathbf{t} + \mathbf{t}')$$

Thus Trans and Sim produce $\mathbf{h}$ from identical distribution as well.

As the output distributions of Trans and Sim are exactly identical, $\varepsilon_{\mathsf{zk}} = 0$.

Correctness We now consider the correctness error of $\mathsf{blDilithium\text{-}QROM\text{-}ID}$ in the sense of Definition 2.3 of [27], which involves the probability of both the prover failing and the verifier failing on a valid output of the prover.

blDilithium-QROM.KeyGen()

$K \leftarrow \{0,1\}^{256}$
$(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} S_\eta^\ell \times S_\eta^k$
$\mathbf{t} \leftarrow \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
$\mathbf{t}_1 \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$
$\mathbf{t}_0 \leftarrow \mathbf{t} - \mathbf{t}_1 \cdot 2^d$
$pk \leftarrow (\mathbf{t}_1, \mathbf{t}_0)$
$sk \leftarrow (\mathbf{s}_1, \mathbf{s}_2, K)$
return (pk, sk)

blDilithium-QROM.BlindPk$(pk = (\mathbf{t}_1, \mathbf{t}_0), \tau)$

$(\mathbf{s}_1', \mathbf{s}_2') \leftarrow \mathsf{G}(pk \| \tau)$
$\mathbf{t}' \leftarrow \mathbf{A}\mathbf{s}_1' + \mathbf{s}_2'$
$\mathbf{t} \leftarrow \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$
$\mathbf{t}_\tau \leftarrow \mathbf{t} + \mathbf{t}'$
$\mathbf{t}_{1,\tau} \leftarrow \mathsf{Power2Round}_q(\mathbf{t}_\tau, d)$
$\mathbf{t}_{0,\tau} \leftarrow \mathbf{t}_\tau - \mathbf{t}_{1,\tau} \cdot 2^d$
$pk_\tau \leftarrow (\mathbf{t}_{1,\tau}, \mathbf{t}_{0,\tau})$
return pk_τ

blDilithium-QROM.Sign$(M, pk = (\mathbf{t}_1, \mathbf{t}_0), sk = (\mathbf{s}_1, \mathbf{s}_2, K), \tau)$

$(\mathbf{s}_1', \mathbf{s}_2') \leftarrow \mathsf{G}(pk \| \tau)$
$\mathbf{s}_{1,\tau} \leftarrow \mathbf{s}_1 + \mathbf{s}_1'$
$\mathbf{s}_{2,\tau} \leftarrow \mathbf{s}_2 + \mathbf{s}_2'$
$\mathbf{t}_\tau \leftarrow \mathbf{A}\mathbf{s}_{1,\tau} + \mathbf{s}_{2,\tau}$
$\mathbf{t}_{1,\tau} \leftarrow \mathsf{Power2Round}_q(\mathbf{t}_\tau, d)$
$\mathbf{t}_{0,\tau} \leftarrow \mathbf{t}_\tau - \mathbf{t}_{1,\tau} \cdot 2^d$
$\kappa \leftarrow 0$
while $(\mathbf{z}, \mathbf{h}) = (\perp, \perp)$ **and** $\kappa \leq 200/(1-\delta)$ **do**
  $\kappa \leftarrow \kappa + 1$
  $\mathbf{y} \xleftarrow{\$} S_{\gamma'-1}^\ell$
  $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$
  $\mathbf{w}_1 \leftarrow \mathsf{HighBits}_q(\mathbf{w}, 2\gamma)$
  $c \leftarrow \mathsf{H}(M \| \mathbf{w}_1 \| \mathbf{t}_{1,\tau})$
  $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_{1,\tau}$
  if $\|\mathbf{z}\|_\infty \geq \gamma' - \beta$ **or** $\left\|\mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_{2,\tau}, 2\gamma)\right\|_\infty \geq \gamma - \beta$ **then** $(\mathbf{z}, \mathbf{h}) \leftarrow (\perp, \perp)$
  else $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_{0,\tau}, \mathbf{w} - c\mathbf{s}_{2,\tau} + c\mathbf{t}_{0,\tau}, 2\gamma)$
return $\sigma = (\mathbf{z}, \mathbf{h}, c)$

blDilithium-QROM.Verify$(M, \sigma = (\mathbf{z}, \mathbf{h}, c), pk_\tau = (\mathbf{t}_{1,\tau}, \mathbf{t}_{0,\tau}))$

$\mathbf{w}_1' \leftarrow \mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_{1,\tau} \cdot 2^d, 2\gamma)$
if $\|\mathbf{z}\|_\infty < \gamma' - \beta$ **and** $c = \mathsf{H}(M \| \mathbf{w}_1 \| \mathbf{t}_{1,\tau})$ **then return** accept
else return reject

Fig. 5. Key generation, blinding, signing, and verification algorithms for blDilithium-QROM with extra information published in the identity public key. This scheme is used for the proofs of unforgeability and unlinkability.

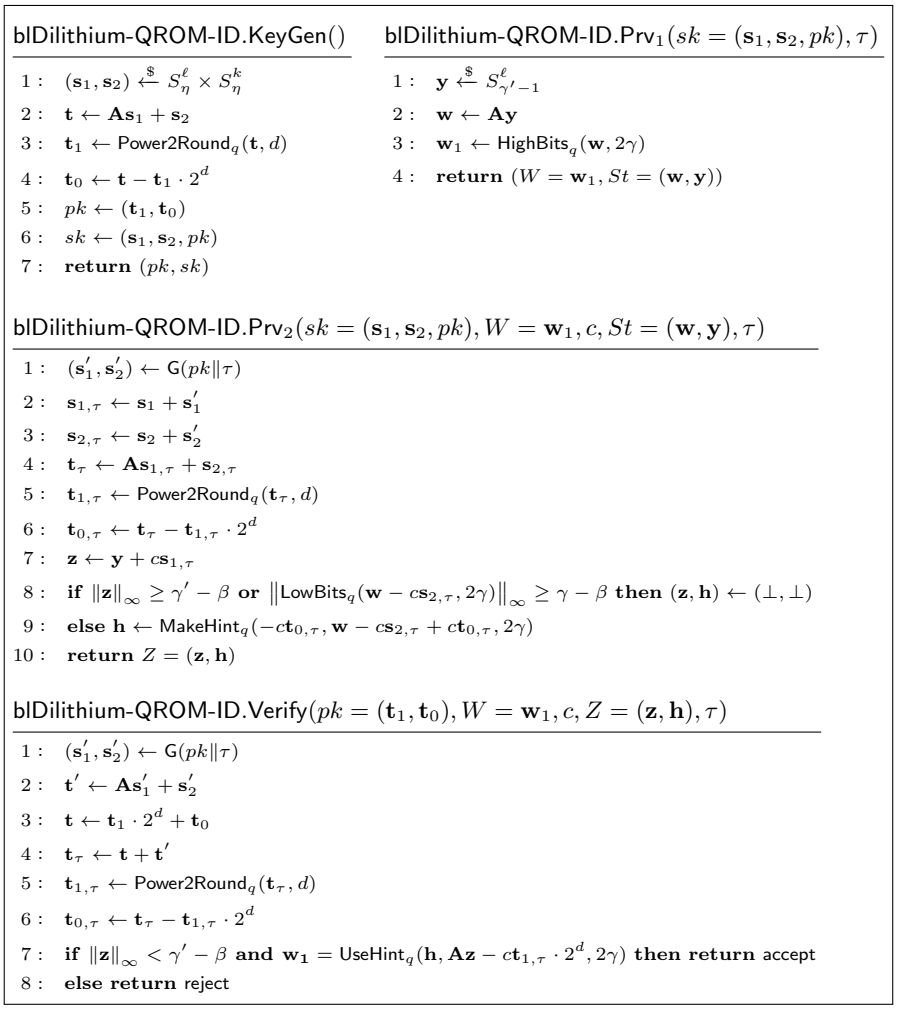


Fig. 6. Key generation, proving, and verification algorithms for an identification scheme (blDilithium-QROM-ID) used in the proof of unforgeability of the blDilithium-QROM signature scheme.

Trans	Sim
1 : $(\mathbf{s}_1', \mathbf{s}_2') \leftarrow \mathsf{G}(pk \| \tau)$	1 : $(\mathbf{s}_1', \mathbf{s}_2') \leftarrow \mathsf{G}(pk \| \tau)$
2 : $\mathbf{s}_{1,\tau} \leftarrow \mathbf{s}_1 + \mathbf{s}_1'$	2 : $\mathbf{t}' \leftarrow \mathbf{A}\mathbf{s}_1' + \mathbf{s}_2'$
3 : $\mathbf{s}_{2,\tau} \leftarrow \mathbf{s}_2 + \mathbf{s}_2'$	3 : $\mathbf{t} \leftarrow \mathbf{t}_1 \cdot 2^d + \mathbf{t}_0$
4 : $\mathbf{t}_\tau \leftarrow \mathbf{A}\mathbf{s}_{1,\tau} + \mathbf{s}_{2,\tau}$	4 : $\mathbf{t}_\tau \leftarrow \mathbf{t} + \mathbf{t}'$
5 : $\mathbf{t}_{1,\tau} \leftarrow \mathsf{Power2Round}_q(\mathbf{t}_\tau, d)$	5 : $\mathbf{t}_{1,\tau} \leftarrow \mathsf{Power2Round}_q(\mathbf{t}_\tau, d)$
6 : $\mathbf{t}_{0,\tau} \leftarrow \mathbf{t}_\tau - \mathbf{t}_{1,\tau} \cdot 2^d$	6 : $\mathbf{t}_{0,\tau} \leftarrow \mathbf{t}_\tau - \mathbf{t}_{1,\tau} \cdot 2^d$
7 : $\mathbf{y} \xleftarrow{\$} S^\ell_{\gamma'-1}$	7 : with probability $1 - \frac{\left\lvert S^\ell_{\gamma'-\beta-1}\right\rvert}{\left\lvert S^\ell_{\gamma'-1}\right\rvert}$ **return** $(\perp, (\perp, \perp))$
8 : $\mathbf{w} \leftarrow \mathbf{A}\mathbf{y}$	8 : $\mathbf{z} \xleftarrow{\$} S^\ell_{\gamma'-\beta-1}$
9 : $\mathbf{w}_1 \leftarrow \mathsf{HighBits}_q(\mathbf{w}, 2\gamma)$	9 : $c \xleftarrow{\$} \mathsf{ChSet}$
10 : $c \xleftarrow{\$} \mathsf{ChSet}$	10 : **if** $\left\|\mathsf{LowBits}_q(\mathbf{A}\mathbf{z} - c\mathbf{t}_\tau, 2\gamma)\right\|_\infty \geq \gamma - \beta$ **then**
11 : $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_{1,\tau}$	11 : **return** $(\perp, (\perp, \perp))$
12 : **if** $\|\mathbf{z}\|_\infty \geq \gamma' - \beta$ **then return** $(\perp, (\perp, \perp))$	12 : $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_{0,\tau}, \mathbf{A}\mathbf{z} - c\mathbf{t}_\tau + c\mathbf{t}_{0,\tau}, 2\gamma)$
13 : **if** $\left\|\mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_{2,\tau}, 2\gamma)\right\|_\infty \geq \gamma - \beta$ **then**	13 : **return** $(c, (\mathbf{z}, \mathbf{h}))$
14 : **return** $(\perp, (\perp, \perp))$	
15 : $\mathbf{h} \leftarrow \mathsf{MakeHint}_q(-c\mathbf{t}_{0,\tau}, \mathbf{w} - c\mathbf{s}_{2,\tau} + c\mathbf{t}_{0,\tau}, 2\gamma)$	
16 : **return** $(c, (\mathbf{z}, \mathbf{h}))$	

Fig. 7. Real and simulated transcripts of the blDilithium-QROM-ID protocol.

Lemma 4. *If* $\max_{s \in S_\eta, c \in \mathsf{ChSet}} \|2cs\|_\infty \leq \beta$, $\max_{t_0 \in S'_{2^d}, c \in \mathsf{ChSet}} \|2ct_0\|_\infty \leq \gamma$, $\beta \ll \gamma'$, *and* $\beta + 1 < 2\beta$, *then* blDilithium-QROM-ID *has correctness error*

$$\delta \approx 1 - \exp\left(-\beta n \left(\frac{\ell}{\gamma'} + \frac{k}{\gamma}\right)\right)$$

Proof. We begin by computing the probability Prv_1 or Prv_2 does not output $(\perp, (\perp, \perp))$. The probability $(\perp, (\perp, \perp))$ is not output in line 12 of Trans is simply the probability it is not output in line 7 of Sim, thus this probability is

$$\frac{\left\lvert S^\ell_{\gamma'-\beta-1}\right\rvert}{\left\lvert S^\ell_{\gamma'-1}\right\rvert} = \left(\frac{2(\gamma'-\beta)-1}{2\gamma'-1}\right)^{n\ell} > \left(1 - \frac{\beta}{\gamma'}\right)^{n\ell} \approx \exp\left(-\frac{\beta n \ell}{\gamma'}\right)$$

as $\beta \ll \gamma'$. On the assumption that the distribution of $\mathbf{A}\mathbf{z} - c(\mathbf{t} + \mathbf{t}') \mod 2\gamma$ is close to uniform when $\mathbf{z} \in S^k_{\gamma'-\beta-1}$ is uniformly sampled, then the probability $(\perp, (\perp, \perp))$ is not output in line 14 of Trans or equivalently line 11 of Sim is

$$\Pr[\mathbf{z} \xleftarrow{\$} S^\ell_{\gamma'-\beta-1} \mid \left\|\mathsf{LowBits}_q(\mathbf{A}\mathbf{z} - c(\mathbf{t} + \mathbf{t}'))\right\|_\infty < \gamma - \beta] \approx \frac{\left\lvert S^k_{\gamma-\beta-1}\right\rvert}{\left\lvert S^k_{\gamma-1}\right\rvert} \approx \exp\left(-\frac{\beta n k}{\gamma}\right)$$

Hence

$$\Pr[\mathbf{y} \xleftarrow{\$} S^\ell_{\gamma'-1}, c \xleftarrow{\$} \mathsf{ChSet} \mid (\mathbf{z}, \mathbf{h}) \neq (\perp, \perp)] \approx \exp\left(-\beta n \left(\frac{\ell}{\gamma'} + \frac{k}{\gamma}\right)\right)$$

Finally, assume $(\mathbf{z}, \mathbf{h}) \neq (\perp, \perp)$. Then blDilithium-QROM-ID.Verify will always accept. Clearly, $\|\mathbf{z}\|_\infty < \gamma' - \beta$. Also, since

$$\mathbf{w} - c(\mathbf{s}_2 + \mathbf{s}_2') = \mathbf{A}\mathbf{z} - c(\mathbf{t}_0 + \mathbf{t}_0') - c(\mathbf{t}_1 + \mathbf{t}_1') \cdot 2^d$$

and $\left\|c(\mathbf{t}_0 + \mathbf{t}_0')\right\|_\infty < \gamma$ and $\left\|\mathsf{LowBits}_q(\mathbf{w} - c(\mathbf{s}_2 + \mathbf{s}_2'), 2\gamma)\right\|_\infty < \gamma - \beta$, by Lemmas 4.1 and 4.2 of [27],

$$\mathsf{UseHint}_q(\mathbf{h}, \mathbf{A}\mathbf{z} - c(\mathbf{t}_1 + \mathbf{t}_1') \cdot 2^d, 2\gamma) = \mathsf{HighBits}_q(\mathbf{w} - c(\mathbf{s}_2 + \mathbf{s}_2'), 2\gamma) = \mathbf{w}_1$$

Hence, blDilithium-QROM-ID has correctness error based solely off of the probability Prv fails, thus

$$\delta \approx 1 - \exp\left(-\beta n \left(\frac{\ell}{\gamma'} + \frac{k}{\gamma}\right)\right)$$

Lossiness We now show that a bounded adversary has trouble distinguishing valid identity public keys as done in Fig 6 and randomly generated identity public keys as done in Fig 8. In addition, given a randomly generated identity public key, any unbounded adversary has only a little more than $1/|\mathsf{ChSet}|$ probability of impersonating the prover. More concretely, we address these two properties as defined in Definition 2.8 of [27].

blDilithium-QROM-ID.LosKeyGen()
1 : $\mathbf{t} \xleftarrow{\$} R_q^k$
2 : $\mathbf{t}_1 \leftarrow \mathsf{Power2Round}_q(\mathbf{t}, d)$
3 : $\mathbf{t}_0 \leftarrow \mathbf{t} - \mathbf{t}_1 \cdot 2^d$
4 : **return** $pk = (\mathbf{t}_1, \mathbf{t}_0)$

Fig. 8. Lossy key generator of blDilithium-QROM-ID.

Lemma 5. *For any adversary $\mathcal{A}$,*

$$\mathbf{Adv}^{\mathsf{LOSS}}_{\mathsf{blDilithium\text{-}QROM\text{-}ID}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{k,\ell,\mathcal{U}}(\mathcal{A})$$

where $\mathcal{U}$ is the uniform distribution over S_η.

Proof. Differentiating between the output of blDilithium-QROM-ID.KeyGen and blDilithium-QROM-ID.LosKeyGen is clearly equivalent to differentiating between MLWE samples with static $\mathbf{A}$ and uniform samples over R_q^k.

Game LOSSY-IMP

$pk_{\mathsf{ls}} = (\mathbf{t}_1, \mathbf{t}_0) \leftarrow$ blDilithium-QROM-ID.LosKeyGen()
$\tau \leftarrow \mathcal{A}()$
$(\mathbf{w}_1, St) \leftarrow \mathcal{A}(pk_{\mathsf{ls}}, \tau)$
$c \leftarrow \mathsf{ChSet}$
$(\mathbf{z}, \mathbf{h}) \leftarrow \mathcal{A}(St, c, \tau)$
return $\mathsf{Verify}(pk_{\mathsf{ls}}, \mathbf{w}_1, c, (\mathbf{z}, \mathbf{h}))$

Fig. 9. LOSSY-IMP game.

Lemma 6. *If* $q \equiv 5 \pmod 8$, $\max_{s \in S_\eta, c \in \mathsf{ChSet}} \|2cs\|_\infty \leq \beta$, $4\gamma + 2^d\beta + 2, 2\gamma' + (2^d - 2)\beta - 2 < \sqrt{q/2}$, *and* $2^d < 4\gamma' + (2^{d+1} - 4)\beta - 4$, *then* blDilithium-QROM-ID *has* $\varepsilon_{\mathsf{ls}}$*-lossy soundness for*

$$\varepsilon_{\mathsf{ls}} \leq \frac{1}{|\mathsf{ChSet}|} + 2|\mathsf{ChSet}|^2 \cdot \left(\frac{(4\gamma' + (2^{d+1} - 4)\beta - 3)^\ell \cdot (8\gamma + 2^{d+1}\beta + 5)^k}{q^k} \right)^n$$

Proof. Let $\mathcal{A}$ be an unbounded adversary executed in the LOSSY-IMP game as shown in Fig 9. We first consider the case where there exist two distinct $(c, (\mathbf{z}, \mathbf{h})), (c', (\mathbf{z}', \mathbf{h}'))$ such that $\mathcal{A}$ is able to impersonate the prover. It follows that $\|\mathbf{z}\|_\infty, \|\mathbf{z}'\|_\infty < \gamma' - \beta$ and

$$\mathbf{w}_1 = \mathsf{UseHint}_q(\mathbf{h}, \mathbf{Az} - (\mathbf{t}_1 + \mathbf{t}_1')c \cdot 2^d, 2\gamma) = \mathsf{UseHint}_q(\mathbf{h}', \mathbf{Az}' - (\mathbf{t}_1 + \mathbf{t}_1')c' \cdot 2^d, 2\gamma)$$

Thus by Lemma 4.1 of [27]

$$\left\| \mathbf{Az} - (\mathbf{t}_1 + \mathbf{t}_1')c \cdot 2^d - \mathbf{w}_1 \cdot 2\gamma \right\|_\infty \leq 2\gamma + 1$$

$$\left\| \mathbf{Az}' - (\mathbf{t}_1 + \mathbf{t}_1')c' \cdot 2^d - \mathbf{w}_1 \cdot 2\gamma \right\|_\infty \leq 2\gamma + 1$$

So by the triangle equality,

$$\left\| \mathbf{A}(\mathbf{z} - \mathbf{z}') - (\mathbf{t}_1 + \mathbf{t}_1') \cdot (c - c') \cdot 2^d \right\|_\infty \leq 4\gamma + 2$$

Hence, for some $\mathbf{u}$ such that $\|\mathbf{u}\|_\infty \leq 4\gamma + 2$,

$$\mathbf{A}(\mathbf{z} - \mathbf{z}' - \mathbf{s}_1' \cdot 2^d(c - c')) + (\mathbf{u} - \mathbf{s}_2' \cdot 2^d(c - c')) = \mathbf{t}_1 \cdot 2^d(c - c')$$

Since $\|2c\mathbf{s}_1\|_\infty \leq \beta$, then $\left\|\mathbf{z} - \mathbf{z}' - \mathbf{s}_1' \cdot 2^d(c - c')\right\|_\infty \leq 2(\gamma' - \beta - 1) + 2^d\beta$ and $\left\|\mathbf{u} - \mathbf{s}_2' \cdot 2^d(c - c')\right\|_\infty \leq 4\gamma + 2 + 2^d\beta$, then by Lemma 4.6 of [27], the above equation is satisfied with probability

$$2|\mathsf{ChSet}|^2 \cdot \left(\frac{(4\gamma' + (2^{d+1} - 4)\beta - 3)^\ell \cdot (8\gamma + 2^{d+1}\beta + 5)^k}{q^k} \right)^n$$

In the case where there is only one c that allows $\mathcal{A}$ to impersonate the prover, then $\mathcal{A}$ only has a $1/|\mathsf{ChSet}|$ probability of winning the LOSSY-IMP game. The Lemma follows from combining probabilities of both these cases.

Min Entropy We finally consider the probability that the $\mathbf{w}_1$ output is distinct for every run of the prover. As Prv_1 is identical between Dilithium-QROM-ID and blDilithium-QROM-ID, Lemma 4.7 of [27] applies directly to this setting. As such, we restate Lemma 4.7 here.

Lemma 7 (Lemma 4.7 of [27]). *If* $2\gamma, 2\gamma' < \sqrt{q/2}$ *and* $\ell \leq k$ *then* blDilithium-QROM-ID *has*

$$\alpha > n\ell \cdot \log\left(\min\left(\frac{q}{(4\gamma+1)(4\gamma'+1)}, 2\gamma'-1\right)\right)$$

bits of min-entropy as defined in Definition 2.6 of [27].

Unforgeability under Chosen Message and Epoch Attack We now arrive at showing the unforgeability of blDilithium-QROM in the context of blinded signature schemes. First observe that $\mathsf{FS}[\mathsf{blDilithium\text{-}QROM\text{-}ID}, \mathsf{H}, \frac{200}{1-\delta}]$ is equivalent to blDilithium-QROM.

The following Theorem is a direct result of preceding Lemmas as well as Theorem 3.1 of [27].

Theorem 3. *Let* $\mathcal{A}$ *be any adversary that makes at most* q_{H} *hash queries and* q_S *signing queries against the unforgeability of* blDilithium-QROM *with parameters as specified in Figure 4. Then there exists an algorithm* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{k,\ell,\mathcal{U}}(\mathcal{B}) + 8(q_{\mathsf{H}}+1)\cdot 2^{-137} + 2^{-2899}$$

where $\mathsf{Time}(\mathcal{A}) \approx \mathsf{Time}(\mathcal{B})$.

Proof. By Theorem 3.1 of [27],

$$\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{LOSS}}_{\mathsf{blDilithium\text{-}QROM\text{-}ID}}(\mathcal{B}) + 8(q_{\mathsf{H}}+1)\varepsilon_{\mathsf{ls}} + \kappa_m q_S \varepsilon_{\mathsf{zk}} + 2^{1-\alpha}$$

By Lemma 5,

$$\mathbf{Adv}^{\mathsf{LOSS}}_{\mathsf{blDilithium\text{-}QROM\text{-}ID}}(\mathcal{B}) = \mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{k,\ell,\mathcal{U}}(\mathcal{B})$$

where $\mathcal{U}$ is the uniform distribution over S_η. By Lemma 6,

$$\varepsilon_{\mathsf{ls}} \leq \frac{1}{|\mathsf{ChSet}|} + 2|\mathsf{ChSet}|^2 \cdot \left(\frac{(4\gamma' + (2^{d+1}-4)\beta - 3)^\ell \cdot (8\gamma + 2^{d+1}\beta + 5)^k}{q^k}\right)^n$$

Substituting in the parameters from Figure 4 and noting that the magnitude of $|\mathsf{ChSet}|$ is $\binom{512}{46} \cdot 2^{46} > 2^{265}$, we get that $\varepsilon_{\mathsf{ls}} \leq 2^{-265} + 2^{-138} \leq 2^{-137}$. Also, by construction, $\kappa_m = \frac{200}{1-\delta} \approx 3681$. By Lemma 3, $\varepsilon_{\mathsf{zk}} = 0$. Finally, by Lemma 7, $2^{1-\alpha} < 2^{-2899}$. The theorem follows directly from these observations.

A.4 blDilithium-QROM Unlinkability

We now proceed to showing that blDilithium-QROM is unlinkable, and begin by demonstrating that it satisfies the independent blinding property.

Lemma 8. *If the MWLE problem is hard, then* blDilithium-QROM *satisfies independent blinding. In particular, for any adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{Ind-Blind}}_{\mathsf{blDilithium\text{-}QROM},t}(\mathcal{A}) \leq 2t\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\mathcal{U},\mathbf{A}}(\mathcal{B})$$

where $\mathcal{U}$ is the uniform distribution over S_η.

Proof. We begin by defining the randBlind function for blDilithium-QROM as identical to BlindPk except that $(\mathbf{s}'_1, \mathbf{s}'_2)$ are uniformly sampled from $S^\ell_\eta \times S^k_\eta$ rather than output from $\mathsf{G}(pk\|\tau)$.

Let G_0 be a game where $\mathcal{A}$ is given independent blindings of a single identity public key and game G_4 be a game where $\mathcal{A}$ is given independent blindings of independent identity public keys.

We first introduce a game G_1 which differs from G_0 in that each output of randBlind, specifically $\mathbf{t}_0 + \mathbf{t}'_i$ for $1 \leq i \leq t$ where $\mathbf{t}'_i$ is an MLWE sample, is replaced with $\mathbf{t}_0 + \tilde{\mathbf{t}}'_i$ for $1 \leq i \leq t$ where $\tilde{\mathbf{t}}'_i$ is randomly and uniformly sampled from R^k_q. Note that any adversary $\mathcal{A}$ that can differentiate between a single sample $\mathbf{t}_0 + \mathbf{t}'$ and $\mathbf{t}_0 + \tilde{\mathbf{t}}'$ can be used to construct an adversary $\mathcal{B}$ that differentiates between plain MLWE samples, as the MLWE challenge can be transformed into an input $\mathcal{A}$ uses by simply adding $\mathbf{t}_0$ to the challenge. As we must replace each of the t samples of $\mathbf{t}_0 + \mathbf{t}'_i$ individually, the triangle equality implies that $\left|Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_1} \Rightarrow 1]\right| \leq t\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\mathcal{U},\mathbf{A}}(\mathcal{B})$.

We now introduce G_2, which is identical to G_1 except that each sample $\mathbf{t}_0 + \tilde{\mathbf{t}}'_i$ for $1 \leq i \leq t$ is replaced with a sample $\tilde{\mathbf{t}}_i$ where $\tilde{\mathbf{t}}_i$ is randomly and uniformly sampled from R^k_q. Recall that $\tilde{\mathbf{t}}'_i$ is randomly and uniformly sampled from R^k_q, thus it is effectively randomly permuting $\mathbf{t}_0$ over R^k_q so G_1 and G_2 are indistinguishable.

In G_3, we now take the reverse step of replacing each $\tilde{\mathbf{t}}_i$ for $1 \leq i \leq t$ in G_2 with $\mathbf{t}_i + (\tilde{\mathbf{t}}_i - \mathbf{t}_i)$. Note $\tilde{\mathbf{t}}_i - \mathbf{t}_i$ is uniformly random over R^k_q. G_3 is clearly equivalent to G_2.

Finally, we make the hop from G_3 to G_4 by replacing each $\mathbf{t}_i + (\tilde{\mathbf{t}}_i - \mathbf{t}_i)$ with $\mathbf{t}_i + \mathbf{t}'_i$ where $\mathbf{t}'_i$ is an MLWE sample for $1 \leq i \leq t$. As $\tilde{\mathbf{t}}_i - \mathbf{t}_i$ is uniformly random over R^k_q, the bound in the difference between G_3 and G_4 is the same as the bound in the difference between G_0 and G_1. Hence by the triangle equality,

$$\left|Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_4} \Rightarrow 1]\right| = \mathbf{Adv}^{\mathsf{Ind-Blind}}_{\mathsf{blDilithium\text{-}QROM},t}(\mathcal{A}) \leq 2t\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\mathcal{U},\mathbf{A}}(\mathcal{B})$$

We now show that blDilithium-QROM permits signing with oracle reprogramming as specified in Definition 4. Let Forge output $(y = c, \sigma = (\mathbf{z}, \mathbf{h}, c))$ where $\mathbf{z}$, $\mathbf{h}$, and c are output from Sim in Figure 7. In addition, let Ext output $M\|\mathsf{UseHint}_q(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_{1,\tau} \cdot 2^d, 2\gamma)\|\mathbf{t}_{1,\tau}$. Since by Lemma 3 blDilithium-QROM is naHVZK given appropriate parameters, then real signatures and the output of

Forge are drawn from the exact same distributions, thus $\delta = 0$. Furthermore, the min entropy of Ext is solely dependent on $\mathbf{w}_1 = \mathbf{Az} - c\mathbf{t}_{1,\tau} \cdot 2^d$, as the adversary has control over M and knowledge of $\mathbf{t}_{1,\tau}$, hence the min entropy of Ext is the same as the min entropy from Lemma 7.

We are now ready to consider unlinkability under chosen message and epoch attack, which follows directly from the preceding Lemmas as well as the Lemmas in Section 3.

Theorem 4. *For any adversary $\mathcal{A}$ that makes q_S signing queries and q_H random oracle queries, there exists an algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{UL-CMEA}}_{\mathsf{blDilithium\text{-}QROM},t}(\mathcal{A}) \leq 2t\mathbf{Adv}^{\mathsf{SA\text{-}MLWE}}_{m,k,\mathcal{U},\mathbf{A}}(\mathcal{B}_1) + q_\mathsf{H}\mathbf{Adv}^{\mathsf{EUF-CMEA}}_{\mathsf{blDilithium\text{-}QROM}}(\mathcal{B}_2) + q_\mathsf{H}q_S 2^{-2899}$$

B A Generic Zero-Knowledge Key-Blinding Scheme

In this section we explain how one of the most unique submissions to NIST's post-quantum standardization effort, Picnic [16], can also be transformed into a key-blinded signature scheme. Picnic improves upon techniques introduced in [20, 25] to construct a signature scheme out of the generic zero-knowledge proof system ZKB++. A secret key in Picnic is the secret key to a symmetric key encryption function, $k \in \{0,1\}^\lambda$. The public key is a pair of values $(x, y) \in \{0,1\}^{2\lambda}$ such that $y = \mathsf{Enc}_k(x)$ for some suitable encryption function Enc.

To sign a message msg, the signer constructs a generic zero-knowledge proof, dependent on msg that they know a key k under which x is encrypted to y. The hash of msg is then mixed into the randomness used to generate this zero-knowledge proof in such a way to result in an existentially unforgeable signature scheme. The zero-knowledge proof itself is based on the 'MPC in the head' methodology proposed in [25], which noted a fundamental connection between zero-knowledge proofs and multi-party computation (MPC).

We present a rough outline of the signature scheme Picnic, and refer to [16] for full details. Consider three separate parties, A, B, and C, who possess private values a, b, and c, respectively. They are using multi-party computation to compute $y = \mathsf{Enc}_k(x)$ for some fixed value x and $k = a \oplus b \oplus c$. In the end, no party will learn k. Indeed, for certain configurations of multi-party computation, even if two out of the three parties collaborate, they cannot learn k.

We can consider the views of each of these parties, consisting of all incoming and outgoing messages as well as all intermediate values in the computation. Because parties can collaborate and still not learn k, the views of two out of the three parties similarly do not contain enough information to decide k. However these views also attest—to a certain extent—to the validity of the computation: One can examine the transcript of the parties views to ascertain that all computations and incoming and outgoing messages were computed correctly.

This realization allows the authors of [25] to show that a secure MPC scheme can be transformed into a zero-knowledge proof. Someone who knows the secret key k can split it up into three parts, a, b, and c, and then run the MPC protocol

'in their head', with each party having a part of k. The respective transcripts of each parties view of the protocol can be committed to, after which a verifier can challenge the prover to reveal the views of two out of the three parties. For a prover to cheat, they must make at least one of the parties misbehave, which can be detected by the verifier with constant probability. But the scheme is actually zero knowledge, as transcripts can easily be simulated by having the party who is not revealed misbehave. By applying the Fiat–Shamir transform, one can obtain a signature scheme: this is Picnic.

Adding a key-blinding functionality to Picnic can be done by encrypting the public key a second time, this time under information derived from the nonce τ. Then the signature will be a zero-knowledge proof of the statement "I know the keys k, k_τ such that $\mathsf{Enc}_{k_\tau}(\mathsf{Enc}_k(x)) = y_\tau$". It's easy to see that this simple mechanism gets us most of the way towards key-blinding functionality. Anyone who knows τ and the previous public key y can derive the new public key y_τ simply by encrypting. Furthermore, the new y_τ is entirely disconnected from y under standard assumptions on the security of the encryption scheme. Anyone who knows τ and y will be unable to sign messages, even though this gives them k_τ as they do not have k and thus cannot construct the signature.

One issue with this system is that while y may be entirely changed by the blinding process, x does not change. If we keep the current system of having the public key system consist of both x and y, with x generated randomly by each user, then the unlinkability of the system is trivially broken, as the x component will not change. To fix this, we must rely on the same technique that we did in the lattice case: x must be a fixed system parameter shared by all users.

$\mathsf{blPicnic.KeyGen}()$	$\mathsf{blPicnic.BlindPk}(y, \tau)$
1 : $k \xleftarrow{\$} \{0,1\}^\lambda$	1 : $k_\tau \leftarrow KDF(y \Vert \tau)$
2 : $y \leftarrow \mathsf{Enc}_k(x)$	2 : $y_\tau \leftarrow \mathsf{Enc}_{k_\tau}(y)$
// x is a system parameter	3 : **return** $pk_\tau = y_\tau$
3 : **return** $(pk, sk) = (y, k)$	

Fig. 10. Key generation and blinding algorithms for blinded Picnic (blPicnic) signature scheme.

The general idea behind the blPicnic Sign and Verify algorithms do not substantially change from Picnic's sign and verification functions. The difference is that the circuit for security is twice as long to perform two encryption functions. As MPC-in-the-head protocols like ZKB++ can handle arbitrary circuits and inputs, this does not change what is possible. In Appendix C we discuss the proof of the unlinkability property.

C blPicnic Proof

Here we outline the proof of unlinkability for blPicnic. For the independent blinding property, we need to show that

$$\mathsf{randBlind}(pk_0; r_1), \ldots, \mathsf{randBlind}(pk_0; r_n)$$

is indistinguishable from

$$\mathsf{randBlind}(pk_1; r_1), \ldots, \mathsf{randBlind}(pk_n; r_n),$$

where for a public key $y = \mathsf{Enc}_k(x)$, we define $randBlind(y, r)$ as just $\mathsf{Enc}_r(y)$. From this it is clear why the proof of unlinkability applies. To distinguish the two distributions is to distinguish many encryptions of the same plaintext versus many encryptions of different plaintexts. This reduces to the indistinguishability of the encryption scheme used. We can proceed by a simple game-hopping argument, where in game i we swap $\mathsf{randBlind}(pk_0; r_i)$ for $\mathsf{randBlind}(pk_i; r_i)$. As long as the ciphertexts are indistinguishable, the hop is justified, and we get that the advantage in breaking the independent blinding property is less than n times the advantage in breaking the indistinguishability of the ciphertext.

The proof of the signing with oracle reprogramming is similar to all Fiat-Shamir schemes. While we do not include all of the details (because we do not explicitly describe the signing protocol), we note that it is essentially the same as the proof for blLegRoast. Proceed through the protocol, but choose which parties will 'cheat' in advance. Then, set the outputs of those parties accordingly so that all checks pass. Reprogram the oracle so that those parties internal states stay secret. All checks will pass and the distribution of the signatures and reprogrammed points is identical.

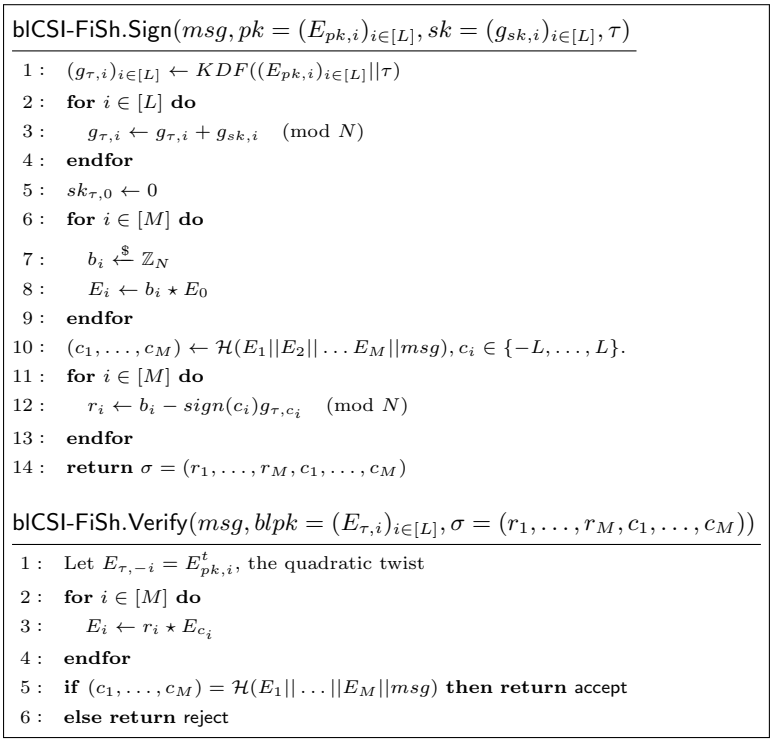
The proof of the unforgeability of the scheme is similarly largely unchanged. The zero-knowledge protocol is meant to work for arbitrary circuits, so encrypting a plaintext twice rather than once is certainly still secure.

However, double encryption does open the possibility of meet-in-the-middle attacks. As the blinded public key now a y_τ value such that $y_\tau = \mathsf{Enc}_{k_\tau}(\mathsf{Enc}_k(x))$, an adversary can sign messages by finding *any* (k, k_τ) such that this relation holds. If the length of the block is ℓ bits, then an adversary can create a list of $2^{\ell/2}$ keys k' and the associated value $\mathsf{Enc}_{k'}(x)$. By then sampling k'_τ values and seeing if $\mathsf{Dec}_{k'_\tau}(y_\tau)$ appears in the list, an adversary can find a secret key that allows them to construct forgeries in time roughly $O(2^{\ell/2})$. Thus, double encryption impacts the security of the underlying one-way function, and the block length ℓ may need to be proportionally increased as a result.

D blCSI-FiSh Proof

The greatest benefit of the CSI-FiSh scheme is that it requires the fewest changes to signing and verification. In this sense it is the most similar to the Ed25519 key-blinding scheme, where the public and secret keys are blinded in a straightforward

way and signing and verification need not change. In Figure 11 we describe the signing and verification procedures. Note that except for the fact that the signing procedure starts by blinding the public key and incorporating the blinding factor into the secret key, it is not changed from in CSI-FiSh.



blCSI-FiSh.Sign$(msg, pk = (E_{pk,i})_{i\in[L]}, sk = (g_{sk,i})_{i\in[L]}, \tau)$

$(g_{\tau,i})_{i\in[L]} \leftarrow KDF((E_{pk,i})_{i\in[L]}||\tau)$
for $i \in [L]$ **do**
  $g_{\tau,i} \leftarrow g_{\tau,i} + g_{sk,i} \pmod N$
endfor
$sk_{\tau,0} \leftarrow 0$
for $i \in [M]$ **do**
  $b_i \xleftarrow{\$} \mathbb{Z}_N$
  $E_i \leftarrow b_i \star E_0$
endfor
$(c_1, \ldots, c_M) \leftarrow \mathcal{H}(E_1||E_2|| \ldots E_M||msg), c_i \in \{-L, \ldots, L\}.$
for $i \in [M]$ **do**
  $r_i \leftarrow b_i - sign(c_i) g_{\tau,c_i} \pmod N$
endfor
return $\sigma = (r_1, \ldots, r_M, c_1, \ldots, c_M)$

blCSI-FiSh.Verify$(msg, blpk = (E_{\tau,i})_{i\in[L]}, \sigma = (r_1, \ldots, r_M, c_1, \ldots, c_M))$

Let $E_{\tau,-i} = E^t_{pk,i}$, the quadratic twist
for $i \in [M]$ **do**
  $E_i \leftarrow r_i \star E_{c_i}$
endfor
if $(c_1, \ldots, c_M) = \mathcal{H}(E_1|| \ldots ||E_M||msg)$ **then return** accept
else return reject

Fig. 11. Key generation and blinding algorithms for blinded CSI-FiSh (blCSI-FiSh) signature scheme.

D.1 Unlinkability

To show that the scheme satisfies unlinkability, we need to prove the independent blinding property and the signing with oracle reprogramming property. We begin with the easier of the two, independent blinding. CSI-FiSh in fact satisfies the strongest possible independent blinding, as the adversaries advantage in distinguishing the two distributions is statistically 0. This is because the the distribution of $\mathsf{randBlind}(pk, r)$ for a uniform r is actually uniform over the entire public key space and independent of pk.

This fact comes from the group action structure. Because we are using a regular group action, the act of sampling a group element $g_{\tau,i} \xleftarrow{\$} \mathbb{Z}_N$ and applying

it to $E_{pk,i}$ provides a uniformly random element of the set, the distribution of which is thus independent of $E_{pk,i}$. So the act of blinding each entry in the public key with a uniform group element causes the entire public key to be perfectly uniformly random and independent, and thus the adversary's advantage in distinguishing the two distributions is 0.

For the signing with oracle reprogramming property, the proof is again a relatively straightforward consequence of the Fiat-Shamir paradigm. By choosing the indices $(c_1, \ldots, c_M)$ that will be returned from $\mathcal{H}$ in advance, we can easily construct a signature that will verify by selecting $r_i \xleftarrow{\$} \mathbb{Z}_N$ and computing $E_i \leftarrow r_i \star E_{c_i}$. It is easy to check the correctness and to see that the fact that our group action is free and transitive will guarantee that the distribution of the signatures generated is statistically identical. Furthermore, the Extract function will return the point $E_1 || \ldots || E_M || msg$, and as the E_i have a distribution uniform over the set (which has size roughly 2^{256}), the resulting min-entropy is 2^{256M}, more than sufficient.

D.2 Unforgeability

The proof of unforgeability is straightforward and not particularly impacted by the addition of blinding. In the CSI-FiSh paper [14], the security of the scheme is proven by showing that the scheme satisfies special soundness, unique responses, large min entropy and satisfies honest-verifier zero knowledge. None of these proofs are changed by the addition of the blinding factor. The blinding factor merely introduces an arbitrary 'offset' for each of the parts of the public key, but this offset does not materially change anything about the security of the scheme, even if an adversary knows the offset.

E blLegRoast Specification and Proof

E.1 Algorithm Description

Here we detail the blinded version of LegRoast or PorcRoast. The scheme mostly resembles the original, but with a few key changes. For the blinded version, we are proving knowledge of a pair $(K, T) \in \mathbb{F}_p$ such that $F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T) = pk_{\tau}$ (more accurately, we are proving a relaxation of this relation, but more on this later). This is done through proving that the error term from equation 1 is equal to zero. LegRoast's error term is very simple, consisting of a single multiplication gate. Our blinded version requires three. This increases the signature length, but also requires some revisions to how the signature scheme works.

$$
\begin{aligned}
E &= \sum_{j=1}^{B} \lambda^{(j)} \left((K + I^{(j)}) \cdot (T + J^{(j)}) \cdot r^{(j)} - o^{(j)} \right) \\
&= K \cdot \left(T \cdot \left(\sum_{j=1}^{B} \lambda^{(j)} r^{(j)} \right) + \sum_{j=1}^{B} \lambda^{(j)} r^{(j)} J^{(j)} \right) \\
&\quad + T \cdot \left(\sum_{j=1}^{B} \lambda^{(j)} I^{(j)} r^{(j)} \right) + \sum_{j=1}^{B} \lambda^{(j)} \left(I^{(j)} J^{(j)} r^{(j)} - o^{(j)} \right), \qquad (1)
\end{aligned}
$$

One issue is that because LegRoast has a single multiplication gate, the output of the gate is publicly known. This allows for some optimizations. As our blinded version has multiple nested multiplication gates, we need to more closely follow the zero-knowledge protocol from [5] that LegRoast uses. This means that the shares of the output gates z are not known, and a correction term Δz must be committed to by the prover. But since the indices $I_e^{(j)}$ are not known until after the prover has committed to the seeds, the correction values Δz are known in a round after the other correction values. To fix this issue, we insert an additional round for the prover. After the λ challenge has been generated, the prover must commit to the Δz correction terms before they get the ϵ challenge values. Other than this, our protocol largely follows that of LegRoast, except with the added values for the two additional multiplication gates used in the computation.

Algorithm 2 blLegRoast.BlindPk

Input: identity public key $pk = (w_1, w_2, \ldots, w_L)$, epoch τ
Output: blinded public key $pk_\tau = (w_{1,\tau}, w_{2,\tau}, \ldots, w_{L,\tau})$
1: $T \leftarrow KDF(pk||\tau)$
2: $v_1 \leftarrow \mathcal{L}_T^\ell(j_1), \ldots, v_L \leftarrow \mathcal{L}_T^\ell(j_L)$
3: $w_{1,\tau} \leftarrow w_1 + v_1 \pmod{\ell}, \ldots, w_{L,\tau} \leftarrow w_L + v_L \pmod{\ell}$
4: **return** pk_τ

E.2 blLegRoast Proof

We now show that blLegRoast satisfies the independent blinding property (Definition 6). We define randBlind by simply using a random $T \in \mathbb{F}_p^\times$ instead of having it determined by the hash function. To show that $\mathbf{Adv}^{\mathsf{Ind-Blind}}(\mathcal{A})$ is small, we need to reduce to the security of $\mathcal{L}$ the ability to distinguish whether $K_1 = K_2 = \cdots = K_n$ or not in the following matrix:

Algorithm 3 blLegRoast.Sign

Input: message m, blinded secret key $sk_\tau = (K, T)$
Output: signature σ

Prover part 1: Generating Pre-processing triples and input shares

Pick a random $\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$.
for $e \in [M]$ **do**
  Sample a root seed $\mathsf{sd}_e \xleftarrow{\$} \{0,1\}^{\lambda}$.
  Build binary tree from sd_e with leaves $\mathsf{sd}_{e,1}, \ldots, \mathsf{sd}_{e,N}$.
  for $i \in [N]$ **do**
    Sample shares: $\left\{ \begin{array}{c} K_{e,i}, T_{e,i}, \\ r_{e,i}^{(1)}, \ldots, r_{e,i}^{(B)}, \\ a_{e,i}^{(1)}, a_{e,i}^{(2)}, a_{e,i}^{(3)}, \\ b_{e,i}^{(1)}, b_{e,i}^{(2)}, b_{e,i}^{(3)}, \\ c_{e,i}^{(1)}, c_{e,i}^{(2)}, c_{e,i}^{(3)}, \\ z_{e,i}^{(1)}, z_{e,i}^{(2)}, z_{e,i}^{(3)} \end{array} \right\} \leftarrow \mathsf{Expand}(\mathsf{sd}_{e,i})$.
    Commit to seed: $\mathsf{com}_{e,i} \leftarrow \mathcal{H}_{com}(\mathsf{salt}, e, i, \mathsf{sd}_{e,i})$
  Compute witness offsets: $\Delta K_e \leftarrow K - \sum_{i=1}^{N} K_{e,i}$, $\Delta T_e \leftarrow T - \sum_{i=1}^{N} T_{i,e}$
  Adjust first shares: $K_{e,1} \leftarrow K_{e,1} + \Delta K_e$, $T_{e,1} \leftarrow T_{e,1} + \Delta T_e$.
  for $k \in [3]$ **do**
    Compute triple: $a_e^{(k)} \leftarrow \sum_{i=1}^{N} a_{e,i}^{(k)}$, $b_e^{(k)} \leftarrow \sum_{i=1}^{N} b_{e,i}^{(k)}$, $c_e^{(k)} \leftarrow a_e^{(k)} \cdot b_e^{(k)}$.
    Compute triple offset: $\Delta c_e^{(k)} \leftarrow c_e^{(k)} - \sum_{i=1}^{N} c_{e,i}^{(k)}$.
    Adjust first share: $c_{e,1}^{(k)} \leftarrow c_{e,1}^{(k)} + \Delta c_e^{(k)}$.
  for $j \in [B]$ **do**
    Compute residue symbol: $s_e^{(j)} \leftarrow \mathcal{L}_0^{\ell}(r_e^{(j)})$, where $r_e^{(j)} \leftarrow \sum_{i=1}^{N} r_{e,i}^{(j)}$.
Set $\sigma_1 \leftarrow \left((\mathsf{com}_{e,i})_{i\in[N]}, (s_e^{(j)})_{j\in[B]}, \Delta K_e, \Delta T_e, (\Delta c_e^{(k)})_{k\in[3]} \right)_{e\in[M]}$.

Verifier part 1: Issuing public key index challenge

Compute challenge hash: $h_1 \leftarrow \mathcal{H}_1(msg, \mathsf{salt}, \sigma_1)$.
Expand hash: $(I_e^{(j)}, J_e^{(j)})_{e\in[M], j\in[B]} \leftarrow \mathsf{Expand}(h_1)$.

Prover part 2: Compute public product values

for $e \in [M]$ **do**
  for $j \in [B]$ **do**
    Compute output value: $o_e^{(j)} \leftarrow (K + I_e^{(j)})(T + J_e^{(j)}) \cdot r_e^{(j)}$.
Set $\sigma_2 \leftarrow (o_e^{(j)})_{e\in[M], j\in[B]}$.

Verifier part 2: Linear combination challenge

Compute challenge hash: $h_2 \leftarrow \mathcal{H}_2(h_1, \sigma_2)$.
Expand the hash $(\lambda_e^{(1)}, \lambda_e^{(2)}, \ldots, \lambda_e^{(B)})_{e\in[M]} \leftarrow \mathsf{Expand}(h_2)$.

Prover part 3: Constructing output shares for multiplication gates

for $e \in [M]$ **do**

Compute first multiplication gate offset:

$$\Delta z_e^{(1)} \leftarrow T \cdot \left(\sum_j \lambda_e^{(j)} r_e^{(j)} \right) - \sum_i z_{e,i}^{(1)}.$$

Compute second multiplication gate offset:

$$\Delta z_e^{(2)} \leftarrow K \cdot \left(\left(\sum_j \lambda_e^{(j)} J_e^{(j)} r_e^{(j)} \right) + T \cdot \sum_j \lambda_e^{(j)} r_e^{(j)} \right) - \sum_i z_{e,i}^{(2)}.$$

Compute final multiplication gate offset:

$$\Delta z_e^{(3)} \leftarrow T \cdot \left(\sum_j \lambda_e^{(j)} I_e^{(j)} r_e^{(j)} \right) - \sum_i z_{e,i}^{(3)}.$$

for $k \in [3]$ **do**

Adjust first shares: $z_{e,1}^{(k)} \leftarrow z_{e,1}^{(k)} + \Delta z_e^{(k)}$.

Set $\sigma_3 \leftarrow (\Delta z_e^{(1)}, \Delta z_e^{(2)}, \Delta z_e^{(3)})_{e \in [M]}$.

Verifier part 3: Multiplication gate challenge

Compute challenge hash: $h_3 \leftarrow \mathcal{H}_3(h_2, \sigma_3)$.

Expand the hash $(\epsilon_e^{(1)}, \epsilon_e^{(2)}, \epsilon_e^{(3)})_{e \in [M]} \leftarrow \mathsf{Expand}(h_3)$.

Prover part 4: Committing to the views of the MPC parties

for $e \in [M]$ **do**

// Views for first multiplication gate

for $i \in [N]$ **do**

Compute shares: $\alpha_{i,e}^{(1)} \leftarrow a_{e,i}^{(1)} + \epsilon_e^{(1)} \cdot T_{e,i}$,

$\beta_{e,i}^{(1)} \leftarrow b_{e,i}^{(1)} + \sum_j \lambda_e^{(j)} r_{e,i}^{(j)}$.

Compute values $\alpha_e^{(1)} \leftarrow \sum_i \alpha_{e,i}^{(1)}$ and $\beta_e^{(1)} \leftarrow \sum_i \beta_{e,i}^{(1)}$.

for $i \in [N]$ **do**

Compute $\gamma_{i,e}^{(1)} \leftarrow \epsilon_e^{(1)} z_{e,i}^{(1)} - c_{e,i}^{(1)} + \alpha_e^{(1)} b_{e,i}^{(1)} + \beta_e^{(1)} a_{e,i}^{(1)}$.

If $i = 1$, set $\gamma_{e,i}^{(1)} \leftarrow \gamma_{e,i}^{(1)} - \alpha_e^{(1)} \cdot \beta_e^{(1)}$.

// Views for second multiplication gate

for $i \in [N]$ **do**

Compute shares: $\alpha_{e,i}^{(2)} \leftarrow a_{e,i}^{(2)} + \epsilon_e^{(2)} \cdot K_{e,i}$,

$\beta_{e,i}^{(2)} \leftarrow b_{e,i}^{(2)} + z_{e,i}^{(1)} + \sum_j \lambda_e^{(j)} J_e^{(j)} r_{e,i}^{(j)}$.

Compute values $\alpha_e^{(2)} \leftarrow \sum_i \alpha_{e,i}^{(2)}$ and $\beta_e^{(2)} \leftarrow \sum_i \beta_{e,i}^{(2)}$.

for $i \in [N]$ **do**

Compute $\gamma_{e,i}^{(2)} \leftarrow \epsilon_e^{(2)} z_{e,i}^{(2)} - c_{e,i}^{(2)} + \alpha_e^{(2)} b_{e,i}^{(2)} + \beta_e^{(2)} a_{e,i}^{(2)}$.

If $i = 1$, set $\gamma_{e,i}^{(2)} \leftarrow \gamma_{e,i}^{(2)} - \alpha_e^{(2)} \cdot \beta_e^{(2)}$.

// Views for final multiplication gate

for $i \in [N]$ **do**

Compute shares: $\alpha_{e,i}^{(3)} \leftarrow a_{e,i}^{(3)} + \epsilon_e^{(3)} \cdot T_{e,i}$

$\beta_{e,i}^{(3)} \leftarrow b_{e,i}^{(3)} + \sum_j \lambda_e^{(j)} I_e^{(j)} r_{e,i}^{(j)}$

Compute values $\alpha_e^{(3)} \leftarrow \sum_i \alpha_{e,i}^{(3)}$, $\beta_e^{(3)} \leftarrow \sum_i \beta_{e,i}^{(3)}$.

for $i \in [N]$ **do**

Compute $\gamma_{e,i}^{(3)} \leftarrow \epsilon_e^{(3)} z_{e,i}^{(3)} - c_{e,i}^{(3)} + \alpha_e^{(3)} b_{e,i}^{(3)} + \beta_e^{(3)} a_{e,i}^{(3)}$.

If $i = 1$, set $\gamma_{e,i}^{(3)} \leftarrow \gamma_{e,i}^{(3)} - \alpha_e^{(3)} \cdot \beta_e^{(3)}$.

Compute output values: $\omega_{e,i} \leftarrow z_{e,i}^{(2)} + z_{e,i}^{(3)} + \sum_j \lambda_e^{(j)} I_e^{(j)} J_e^{(j)} r_{e,i}^{(j)}$

If $i = 1$, set $\omega_{e,i} \leftarrow \omega_{e,i} - \sum_j \lambda_e^{(j)} o_e^{(j)}$.

Set $\sigma_4 \leftarrow \left((\alpha_e^{(k)}, \beta_e^{(k)}, (\alpha_{e,i}^{(k)}, \beta_{e,i}^{(k)}, \gamma_{e,i}^{(k)})_{i \in [N]})_{k \in [3]}, (\omega_{e,i})_{i \in [N]} \right)_{e \in [M]}$.

Verifier Part 4: Challenge on the sacrificing protocol

Compute challenge hash $h_4 \leftarrow \mathcal{H}_4(h_3, \sigma_4)$

Expand hash $(\bar{i}_e)_{e \in [M]} \leftarrow \mathsf{Expand}(h_4)$, where each $\bar{i}_e \in [N]$.

Prover Part 5: Opening the views of the sacrificing protocol

for $e \in [M]$ **do**

Set $\mathsf{seeds}_e \leftarrow \{\log_2(N) \text{ nodes needed to compute } \mathsf{sd}_{e,i} \text{ for } i \in [N] \setminus \{\bar{i}_e\}\}$.

return $\sigma = \left\{ \begin{array}{c} \mathsf{salt}, h_1, h_4, \\ \left(\Delta K_e, \Delta T_e, o_e^{(1)}, \ldots, o_e^{(B)} \right)_{e \in [M]} \\ \left(\alpha_e^{(k)}, \beta_e^{(k)}, \gamma_e^{(k)}, \Delta z_e^{(k)} \right)_{e \in [M], k \in [3]} \\ \left(\mathsf{seeds}_e, \mathsf{com}_{e,\bar{i}} \right)_{e \in [M]} \end{array} \right\}$

Algorithm 4 blLegRoast.Verify

Input: Blinded public key $pk_\tau = (w_{1,\tau}, \ldots, w_{L,\tau})$, signature σ
Output: accept or reject

Parse signature σ as

$$\sigma = \left\{ \begin{array}{c} \mathsf{salt}, h_1, h_4, \\ \left(\Delta K_e, \Delta T_e, o_e^{(1)}, \ldots, o_e^{(B)}\right)_{e\in[M]} \\ \left(\alpha_e^{(k)}, \beta_e^{(k)}, \gamma_e^{(k)}, \Delta z_e^{(k)}\right)_{e\in[M],k\in[3]} \\ \left(\mathsf{seeds}_e, \mathsf{com}_{e,\bar{i}}\right)_{e\in[M]} \end{array} \right\}$$

Compute $h_2 \leftarrow \mathcal{H}_2\left(h_1, (o_e^{(j)})_{e\in[M],j\in[B]}\right)$.
Compute $h_3 \leftarrow \mathcal{H}_3\left(h_2, (\Delta z_e^{(k)})_{e\in[M],k\in[3]}\right)$.
Expand challenge hash 1: $(I_e^{(j)}, J_e^{(j)})_{e\in[M],j\in[B]} \leftarrow \mathsf{Expand}(h_1)$.
Expand challenge hash 2: $(\lambda_e^{(1)}, \lambda_e^{(2)}, \ldots, \lambda_e^{(B)})_{e\in[M]}\mathsf{Expand}(h_2)$.
Expand challenge hash 3: $(\epsilon_e^{(1)}, \epsilon_e^{(2)}, \epsilon_e^{(3)})_{e\in[M]} \leftarrow \mathsf{Expand}(h_3)$.
Expand challenge hash 4: $(\bar{i}_e)_{e\in[M]} \leftarrow \mathsf{Expand}(h_4)$.
for $e \in [M]$ **do**
    Use seeds_e to compute $\mathsf{sd}_{e,i}$ for $i \in [N] \setminus \{\bar{i}\}$.
    for $i \in [N]$ **do**
        Sample shares: $\left\{ \begin{array}{c} K_{e,i}, T_{e,i}, \\ r_{e,i}^{(1)}, \ldots, r_{e,i}^{(B)}, \\ a_{e,i}^{(1)}, a_{e,i}^{(2)}, a_{e,i}^{(3)}, \\ b_{e,i}^{(1)}, b_{e,i}^{(2)}, b_{e,i}^{(3)}, \\ c_{e,i}^{(1)}, c_{e,i}^{(2)}, c_{e,i}^{(3)}, \\ z_{e,i}^{(1)}, z_{e,i}^{(2)}, z_{e,i}^{(3)} \end{array} \right\} \leftarrow \mathsf{Expand}(\mathsf{sd}_{e,i})$.
        if $i = 1$ **then**
            Adjust shares: $K_{e,1} \leftarrow K_{e,1} + \Delta K_e$, $T_1 \leftarrow T_1 + \Delta T_e$
            for $k \in [3]$ **do**
                $c_{e,1}^{(k)} \leftarrow c_{e,1}^{(k)} + \Delta c_e^{(k)}$, $z_{e,1}^{(k)} \leftarrow z_{e,1}^{(k)} + \Delta z_e^{(k)}$.
        Recompute commitments $\mathsf{com}_{e,i} \leftarrow \mathcal{H}_{com}(\mathsf{salt}, e, i, \mathsf{sd}_{e,i})$.
    // First multiplication gate
        Recompute shares: $\overline{\alpha}_{e,i}^{(1)} \leftarrow a_{e,i}^{(1)} + \epsilon_e^{(1)} \cdot T_{e,i}$, $\overline{\beta}_{e,i}^{(1)} \leftarrow b_{e,i}^{(1)} + \sum_j \lambda_e^{(j)} r_{e,i}^{(j)}$.
    // Second multiplication gate
        $\overline{\alpha}_{e,i}^{(2)} \leftarrow a_{e,i}^{(2)} + \epsilon_e^{(2)} \cdot K_{e,i}$, $\overline{\beta}_{e,i}^{(2)} \leftarrow b_{e,i}^{(2)} + z_{e,i}^{(1)} + \sum_j \lambda_e^{(j)} J_e^{(j)} r_{e,i}^{(j)}$.
    // Final multiplication gates
        $\overline{\alpha}_{e,i}^{(3)} \leftarrow a_{e,i}^{(3)} + \epsilon_e^{(3)} \cdot T_{e,i}$, $\overline{\beta}_{e,i}^{(3)} \leftarrow b_{e,i}^{(3)} + \sum_j \lambda_e^{(j)} I_e^{(j)} r_{e,i}^{(j)}$.

for $k \in [3]$ **do**
  Recompute share: $\overline{\gamma}_{e,i}^{(k)} \leftarrow \epsilon_e^{(k)} z_{e,i}^{(k)} - c_{e,i}^{(k)} + \alpha_e^{(k)} b_{e,i}^{(k)} + \beta_e^{(k)} a_{e,i}^{(k)}$
  If $i = 1$ set $\overline{\gamma}_{e,1}^{(k)} \leftarrow \overline{\gamma}_{e,1}^{(k)} - \alpha_e^{(k)} \beta_e^{(k)}$.
Recompute output: $\overline{\omega}_{e,i} \leftarrow z_{e,i}^{(2)} + z_{e,i}^{(3)} + \sum_j \lambda^{(j)} I_e^{(j)} J_e^{(j)} r_{e,i}^{(j)}$.
If $i = 1$, set $\overline{\omega}_{e,1} \leftarrow \overline{\omega}_{e,1} - \sum_j \lambda_e^{(j)} o_e^{(j)}$.
for $k \in [3]$ **do**
  Compute missing shares: $\overline{\alpha}_{e,\bar{i}}^{(k)} \leftarrow \alpha_e^{(k)} - \sum_{i \neq \bar{i}} \overline{\alpha}_{e,i}^{(k)}$
  $\overline{\beta}_{e,\bar{i}}^{(k)} \leftarrow \beta_e^{(k)} - \sum_{i \neq \bar{i}} \overline{\beta}_{e,i}^{(k)}$
  Compute missing check value share: $\overline{\gamma}_{e,\bar{i}}^{(k)} \leftarrow -\sum_{i \neq \bar{i}} \overline{\gamma}_{e,i}^{(k)}$.
Recompute missing output share: $\overline{\omega}_{e,\bar{i}} \leftarrow -\sum_{i \neq \bar{i}} \overline{\omega}_{e,i}$.
for $j \in [B]$ **do**
  Recompute residuosity symbols: $\overline{s}_e^{(j)} \leftarrow \mathcal{L}_0^\ell(o_e^{(j)}) - \mathsf{pk}_{I_e^{(j)}}$.

Check 1:

$$h_1 = \mathcal{H}_1(msg, \mathsf{salt}, ((\mathsf{com}_{e,i})_{i\in[N]}, (\overline{s}_e^{(j)})_{j\in[B]}, \Delta K_e, \Delta T_e, (\Delta c_e^{(k)})_{k\in[3]})_{e\in[M]})$$

Check 2:

$$h_4 = \mathcal{H}_4\left(h_3, \left((\alpha_e^{(k)}, \beta_e^{(k)}, (\overline{\alpha}_{e,i}^{(k)}, \overline{\beta}_{e,i}^{(k)}, \overline{\gamma}_{e,i}^{(k)})_{i\in[N]})_{k\in[3]} (\omega_{e,i})_{i\in[N]}\right)_{e\in[M]}\right)$$

Output accept if both checks pass, fail otherwise.

$$\begin{bmatrix} F_{\mathcal{I}}^\ell(K_1) + F_{\mathcal{J}}^\ell(T_1) \\ F_{\mathcal{I}}^\ell(K_2) + F_{\mathcal{J}}^\ell(T_2) \\ \vdots \\ F_{\mathcal{I}}^\ell(K_n) + F_{\mathcal{J}}^\ell(T_n) \end{bmatrix} = \begin{bmatrix} \mathcal{L}_{K_1}^\ell(i_1) + \mathcal{L}_{T_1}^\ell(j_1) & \dots & \mathcal{L}_{K_1}^\ell(i_L) + \mathcal{L}_{T_1}^\ell(j_L) \\ \mathcal{L}_{K_2}^\ell(i_1) + \mathcal{L}_{T_2}^\ell(j_1) & \dots & \mathcal{L}_{K_2}^\ell(i_L) + \mathcal{L}_{T_2}^\ell(j_L) \\ \vdots & \ddots & \vdots \\ \mathcal{L}_{K_n}^\ell(i_1) + \mathcal{L}_{T_n}^\ell(j_1) & \dots & \mathcal{L}_{K_n}^\ell(i_L) + \mathcal{L}_{T_n}^\ell(j_L) \end{bmatrix}.$$

The natural property of $\mathcal{L}$ to reduce to is its security as a PRF. If, for each key T_i the result on the inputs $\{i_m\}$ for $m \in [L]$ is indistinguishable from random, then each row of the matrix is indistinguishable from uniformly random, independent of whether the K_i values are all the same or not. This means that rather than relying on a search version of the problem that defines the security of the Legendre PRF, we require a decisional version.

Definition 9 (Decisional Fixed Input Power Residue Symbol Problem). *Let p be an odd prime and ℓ be a positive integer with $\ell \mid p-1$. Let $\mathcal{J} = (j_1, j_2, \dots, j_L) \in \mathbb{F}_p^L$. Let $\mathcal{O}_{\mathsf{Pow}}$ be an oracle that, when queried samples a random $T \xleftarrow{\$} \mathbb{F}_p$ and returns $F_{\mathcal{J}}^\ell(T)$. Let $\mathcal{O}_{\mathsf{Ran}}$ be an oracle that returns a uniform element form $\mathbb{Z}_\ell^L$. The* Decisional Fixed Input Power Residue Symbol Problem *is, given $p, \ell, \mathcal{J}, L$ as input distinguish the two oracles with non-negligible probability. We*

write $\mathbf{Adv}^{\mathsf{Ind-PRF}}(\mathcal{A})$ *to denote*

$$\left|\Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Pow}}}] - \Pr[1 \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Ran}}}]\right|.$$

We note that the security of "plain" LegRoast does not depend on such a property. It requires only a search version of the problem, where $\mathcal{A}$ is required to actually find the corresponding T value. The security of the decisional variant and its relation to the search variant is still conjectural. Damgård's original 1988 paper [17] that proposed the use of the Legendre function as a PRF investigates statistical properties with respect to standard randomness tests, but this does not constitute a serious cryptanalytic effort. Grassi et al.'s 2016 paper [21] is largely responsible for the renewed interest in the Legendre function as a PRF. They defined a decisional variant similar to our own, except allowing for adaptive queries. However, later cryptanalytic works [10, 26] focused on the search variant. Security of our scheme relies on the decisional variant; significantly more research on the hardness of the decisional variant is needed before this scheme can be confidently used.

Lemma 9. *If the Legendre PRF is pseudo-random, then* blLegRoast *satisfies the independent blinding property (Definition 6). In particular, for any adversary* $\mathcal{A}$ *that distinguishes a random oracle from the power residue symbols with advantage* $\mathbf{Adv}^{\mathsf{Ind-PRF}}(\mathcal{A})$ *(defined above), we have that* $\mathbf{Adv}^{\mathsf{Ind-Blind}}_{\mathsf{blLegRoast}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\mathsf{Ind-PRF}}(\mathcal{A})$.

Proof. Let game G_0 be one in which $\mathcal{A}$ is given independent blindings of the same public key, and game G_3 be where $\mathcal{A}$ is given independent blindings of independent public keys. We will introduce intermediate games G_1 and G_2, and bound the difference in probability between subsequent hops.

In game G_1, we replace the correct values from $F^{\ell}_{\mathcal{J}}(T_i)$ with uniformly random outputs in $\mathbb{Z}_{\ell}^{L}$ for $i \in [n]$. Since the T_i values are uniformly random already, this replacement causes at most a difference $\mathbf{Adv}^{\mathsf{Ind-PRF}}(\mathcal{A})$, that is $|\Pr[1 \leftarrow \mathcal{A}^{G_0}] - \Pr[1 \leftarrow \mathcal{A}^{G_1}]| \leq \mathbf{Adv}^{\mathsf{Ind-PRF}}$.

As $F^{\ell}_{\mathcal{I}}(K)$ is now being added to uniformly random values, we can replace it with whatever we like and cause no change in the distribution. Thus, for game G_2 we replace each the n instances of $F^{\ell}_{\mathcal{I}}(K)$ by drawing a random K_i and using $F^{\ell}_{\mathcal{I}}(K_i)$ instead. Again, this does not change the distribution of what the adversary has access to, so $|\Pr_{G_1}[1 \leftarrow \mathcal{A}] - \Pr_{G_2}[1 \leftarrow \mathcal{A}]| = 0$.

Finally, for game G_3, we swap the $F^{\ell}_{\mathcal{J}}(T_i)$ values *back* to being honestly generated from T_i, instead of just uniformly random. At this point we have correctly generated $F^{\ell}_{\mathcal{I}}(K_i)$ and $F^{\ell}_{\mathcal{J}}(T_i)$ values, with all K_i uniform and independent. This again introduces a difference bounded by $\mathbf{Adv}^{\mathsf{Ind-PRF}}$. Summing up the differences across all games, we obtain

$$\left|\Pr_{G_0}[1 \leftarrow \mathcal{A}] - \Pr_{G_3}[1 \leftarrow \mathcal{A}]\right| \leq 2 \cdot \mathbf{Adv}^{\mathsf{Ind-PRF}}(\mathcal{A})$$

□

It remains to be shown that the signing with oracle reprogramming property is satisfied and the the scheme is existentially unforgeable. In LegRoast, the proof of existential unforgeability has a fairly straightforward structure. First, the authors show that an adversary $\mathcal{A}$ that attacks the existential unforgeability under chosen message attack can be used to construct an adversary $\mathcal{R}$ that makes no signing queries. This is done by essentially proving the signing with oracle reprogramming property that we defined in Section 3. Then to justify the key-only security, they show that with high probability, an adversary that has successfully constructed a signature has, with all but negligible probability, queried a witness to a β-relaxation of the PRF relation. Finally, they show that, with all but negligible probability, the only witness to such a relaxation is the actual secret key K.

We take a similar approach for the blinded version of LegRoast. The proof that an EU-CMA adversary implies a EU-KO one also works as a proof of the signing with oracle reprogramming property, and closely mirrors the original proof. The proof showing that a key only adversary can be used to recover a witness for the relaxed relation is also similar, with appropriate modifications made for the additional multiplication gates and the extra round. Finally, the proof that the only witness of the relaxation is the original (K, T) proceeds similarly, with a small modification needed to account for the flexibility in having the T parameter. We will present these proofs in the opposite order, starting by talking about the underlying relation and building towards the proof of the EU-CMA security.

Underlying (relaxed) relation The security of the blinded LegRoast scheme depends on the ability of the adversary to find values K and T such that $F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T) = blpk$. In the signature scheme, because of a random subset of the indices of the public key are verified, we need to consider a β-relaxation of finding such a K and T, where only 'enough' of the indices match. We adapt Definitions 1 and 2 from [12] to our situation.

Definition 10 (Additive ℓth-power residue PRF relation). *For an odd prime p, a positive integer $\ell \mid p-1$ and lists $\mathcal{I}, \mathcal{J} \in \mathbb{Z}_p^L$ we define the additive ℓth power residue PRF relation $R^{+}_{\mathcal{L}^{\ell}}$ with output length L as*

$$R^{+}_{\mathcal{L}^{\ell}} = \{(F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T), (K, T)) \in \mathbb{Z}_{\ell}^L \times \mathbb{F}_p \times \mathbb{F}_p\}.$$

Definition 11 (Additive β-approximate PRF relation). *For $\beta \in [0, 1]$, an odd prime p, a positive integer $\ell \mid p-1$, and lists $\mathcal{I}, \mathcal{J} \in \mathbb{Z}_p^L$ define the additive β-approximate PRF relation $R^{+}_{\beta\mathcal{L}^{\ell}}$ with output length L as*

$$R^{+}_{\beta\mathcal{L}^{\ell}} = \{(s, (K, T)) \in \mathbb{Z}_{\ell}^L \times \mathbb{F}_p \times \mathbb{F}_p \mid \exists a \in \mathbb{Z}_{\ell} : d(s + (a, \ldots, a), F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T)) \leq \beta L\},$$

where $d(\cdot, \cdot)$ denotes the Hamming distance.

We can then adapt Theorem 1 from [12] to our situation.

Theorem 5. *Let $\mathcal{B}(n,q)$ denote the binomial distribution with n samples each with success probability q. Let $\mathcal{C}(n,q,m)$ denote the cumulative distribution function with at least m successes, i.e., $\mathcal{C}(n,q,m) = \Pr[B(n,q) \geq m]$. Take $K, T \in \mathbb{F}_p$ and $s = F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T)$. Then with probability at least*

$$1 - \ell p^2 \cdot \mathcal{C}\left(L, \frac{1}{\ell} + \frac{1}{\sqrt{p}} + \frac{2}{p}, (1-\beta)L\right)$$

over the choice of $\mathcal{I}, \mathcal{J}$, there exists only one witness for $s \in R^{+}_{\beta\mathcal{L}^{\ell}}$, which is (K,T), the witness for the exact relation $R^{+}_{\mathcal{L}^{\ell}}$.

To prove the theorem we will require Lemma 1 from [12].

Lemma 10. *Let p be a prime and $\ell \mid p-1$. For any $K, K' \in \mathbb{F}_p$ with $K \neq K'$, and $a \in \mathbb{Z}_{\ell}$, we have*

$$\Pr_{i \xleftarrow{\$} \mathbb{F}_p}[\mathcal{L}^{\ell}(K+i) = \mathcal{L}^{\ell}(K'+i) + a] \leq \frac{1}{\ell} + \frac{1}{\sqrt{p}} + \frac{2}{p}.$$

Proof (of Theorem 5). For any $K', T', j \in \mathbb{F}_p$ with $K' \neq K$, and any $a \in \mathbb{Z}_{\ell}$, we let $a' = \mathcal{L}^{\ell}(T'+j) - \mathcal{L}^{\ell}(T+j) + a$. Then by Lemma 10 we have that the probability, over the choice i that $\mathcal{L}^{\ell}(K+i) = \mathcal{L}^{(}K'+i) + a'$ is bounded. Rearranging the terms of a' we have a bound on the probability that $\mathcal{L}^{\ell}(K+i) + \mathcal{L}(T+j) = \mathcal{L}^{\ell}(K'+i) + \mathcal{L}^{\ell}(T'+j) + a$. As each of the i values is sampled independently, we get that the probability that for a tuple (K', T', a) we have that $d(F^{\ell}_{\mathcal{I}}(K') + F^{\ell}_{\mathcal{J}}(T'), F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T) + (a, \ldots, a)) \leq \beta L$ is

$$C(L, 1/\ell + 1/\sqrt{p} + 2/p, (1-\beta)L).$$

This is true for any $K' \neq K$, and any T' or a. There are $(p-1)$ choices for K', p choices for T', and ℓ choices for a. So the probability that there exists such a K', T', and a is upper bounded by the previous probability multiplied by $\ell(p-1)p$, which we replace with ℓp^2 for simplicity.

Relaxed relation implies key-only security

Theorem 6. *Let q_{com}, q_1, q_2, q_3, and q_4 be the number of queries that an adversary $\mathcal{A}$ makes to random oracles $\mathcal{H}_{com}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$, and $\mathcal{H}_4$ respectively. Fix a constant $\beta \in \{0,1\}$. If $\mathcal{A}$ succeeds in breaking the existential unforgeability of blLegRoast with advantage a then there exists an adversary $\mathcal{R}$ capable of finding a β-approximate witness for blpk with probability at least*

$$a - \frac{MN(q_{sd} + q_1 + q_2 + q_3 + q_4)^2}{2^{2\lambda}} - \Pr[X + Y + Z + W = M]$$

where X is a r.v. distributed as the $\max(X_1, \ldots, X_{q_1})$, with each X_i distributed as $\mathcal{B}(M, (1-\beta)^B)$, and Y, Z, and W defined similarly, but with:

– Y_i *ranging from* $i = 1$ *to* q_2 *and* Y_i *distributed as* $\mathcal{B}(M - X, 1/p)$,
– Z_i *ranging from* $i = 1$ *to* q_3 *and* Z_i *distributed as* $\mathcal{B}(M - X - Y, 3/p)$,
– W_i *ranging from* $i = 1$ *to* q_4 *and* W_i *distributed as* $\mathcal{B}(M - X - Y - Z, 1/N)$.

Proof. We define how the reduction algorithm $\mathcal{R}$ will operate. To begin with, $\mathcal{R}$ is provided $s = F^{\ell}_{\mathcal{I}}(K) + F^{\ell}_{\mathcal{J}}(T)$ which is passed along to $\mathcal{A}$ as the blinded public key. $\mathcal{R}$ will maintain $\mathcal{H}_{com}, \mathcal{H}_1\mathcal{H}_2, \mathcal{H}_3$, and $\mathcal{H}_4$ as random oracles. Typical to random oracle proofs, $\mathcal{R}$ will employ the 'lazy sampling' methodology, and respond consistently when given repeated queries and otherwise sample a random output and provide it to $\mathcal{A}$, maintaining a table of all inputs and outputs. In general, the sampled output is entirely uniform, with some small exceptions:

– The same value is never returned twice (no collisions).
– When the adversary makes a query to $\mathcal{H}_1$ of the form $\Big((\mathsf{com}_{e,i})_{i\in[N]}, \dots\Big)_{e\in[M]}$, add the $\mathsf{com}_{e,i}$ values to a list of values to never return for new queries. This means that $\mathcal{A}$ cannot cannot provide a commitment and then later find an opening to that commitment.
– Do the same thing for queries to $\mathcal{H}_2$ of the form (h_1, σ_2) (do not return h_1), queries to $\mathcal{H}_3$ of the form (h_2, σ_3) (do not return h_2), and queries to $\mathcal{H}_4$ of the form (h_3, σ_4) (do not return h_3). This ensures that the adversary is forced to adhere to the 'correct' order of the Fiat-Shamir paradigm.

Note that $\mathcal{R}$ clearly runs in time roughly the same as $\mathcal{A}$. We will show that, if $\mathcal{A}$ succeeds then with high probability, embedded into $\mathcal{A}$'s queries to the random oracles is enough information to recover a witness (K', T').

The remainder of the proof works very similarly as in [12]. We call specific attention to the cases where the proof differs.

Extracting a β-relaxed witness. To recover a witness, $\mathcal{R}$ looks at the queries made to the hash function. Let $\mathcal{Q}_i$ denote the set of query-responses to oracle $\mathcal{H}_i$. For each query $\sigma_1 = ((\mathsf{com}_{e,i})_{i\in[N]}, \dots, \Delta K_e, \Delta T_e, \dots)_{e\in[M]}$ to $\mathcal{H}_1$, check and see if each $\mathsf{com}_{e,i}$ is the output for a query $(\mathsf{salt}, e, i, \mathsf{sd}_{e,i})$ to $\mathcal{H}_{com}$. If so, then by expanding $sd_{e,i}$ to recover $K_{e,i}$ for each e, i, summing them together over i and adding in ΔK_e, then doing the same for T, we have a candidate witness. For each query to $\mathcal{H}_1$ where this is possible, we maintain a table $\mathcal{T}_i$ of inputs. The table $\mathcal{T}_i$ is indexed by a query σ_1 and the round e and contains the values $K_e, T_e, (r^{(j)}_{e,i})_{i\in[N],j\in[B]}, (a^{(k)}_{e,i}, b^{(k)}_{e,i}, c^{(k)}_{e,i}, z^{(k)}_{e,i})_{i\in[N],k\in[3]}$. Our task is to show that if no such candidate witness can be constructed, then it is only with negligible probability that $\mathcal{A}$ can create a valid signature.

Cheating in the First Phase. Let $\sigma_{best_1}, h_{best_1}$ be the best query-response pair from $\mathcal{H}_1$ that $\mathcal{A}$ receives. 'Best' here refers to maximizing the probability that $\mathcal{A}$ can cheat from this query because the chosen indices that result from the output 'line up' in a way favourable to the adversary. For a query-response pair $((msg, \mathsf{salt}, \sigma_1), h_1)$ to $\mathcal{H}_\infty$, define $G_1(\sigma_1, h_1 = \{I^{(j)}_e\}_{e\in[M],j\in[B]})$ as the rounds $e \in [M]$ for which:

– An entry at index $[\sigma_1, e]$ exist in $\mathcal{T}_i$.
– $\mathcal{L}^\ell((K_e + I_e^{(j)})(T_e + J_e^{(j)})r_e^{(j)}) = s_e^{(j)} + pk_{I_e^{(j)}, J_e^{(j)}}$ for all $j \in [B]$.

We then rely on the following lemma from [12]:

Lemma 11. *If a witness (K, T) cannot be recovered from the queries to $\mathcal{H}_1$ then for any positive integer x, we have that for all query-response pairs to $\mathcal{H}_1$*

$$\Pr[\#G_1(\sigma_1, e) > x] \leq \Pr[X > x]$$

where X is a random variable distributed as $\max(X_1, X_2, \ldots, X_{q_1})$ where each X_i is i.i.d. $\mathcal{B}(M, (1-\beta)^B)$.

Cheating in the second phase. For the second round, any 'functional' (could possibly verify) query-response pair $((h_1, \sigma_2 = (o_e^{(j)})_{e\in[M], j\in[B]}), (\lambda_e^{(j)})_{e\in[M], j\in[B]})$ we define the set of good rounds $G(h_1, \sigma_2, h_2)$ as:

– $\emptyset$ if h_1 is not the output of a query σ_1 to $\mathcal{H}_1$. Otherwise this cannot lead to a valid signature. Let $\sigma_1 = (\ldots, (s_e^{(j)})_{j\in[B]}, \ldots)$.
– $\emptyset$ if there is an index (e, j) such that $\mathcal{L}^\ell(o_e^{(j)}) \neq s_e^{(j)} + pk_{I_e^{(j)}, J_e^{(j)}}$. If there is, then the signature will not verify, because the check on h_1 will fail when reconstructing the s values.
– The values $e \in [M]$ for which the following procedure passes:
  • Using the associated σ_1, recover the inputs from $\mathcal{T}[\sigma_1, e]$.
  • With these inputs, as well as the $I_e^{(j)}, J^{(e)}, o_e^{(j)}$, and $\lambda_e^{(j)}$ values, calculate if the error term E found in Equation 1 is equal to 0.

We once again must establish that if a witness for the relation cannot be recovered from the inputs, then the size of $G_2(h_1, \sigma_2, h_2)$ is bounded. Obviously we must be in the third case, or else $G_2(h_1, \sigma_2, h_2)$ is empty. There are two possibilities:

– A given index $e \in [M]$ is also in $G_1(\sigma_1, h_1)$.
– If the index in not in $G_1(\sigma_1, h_1)$, then it must be due to the fact that there is an index $j \in [B]$ such that $\mathcal{L}^\ell((K_e + I_e^{(j)})(T_e + J_e^{(j)})r_e^{(j)}) \neq s_e^{(j)} + pk_{I_e^{(j)}, J_e^{(j)}}$. But since we have that $\mathcal{L}^\ell(o_e^{(j)}) = s_e^{(j)} + pk_{I_e^{(j)}, J_e^{(j)}}$, this means that the same index has the property that $\mathcal{L}^\ell((K_e + I_e^{(j)})(T_e + J_e^{(j)})r_e^{(j)}) \neq o_e^{(j)}$. This in turn means that the error term is a non-zero linear function in the $\lambda_e^{(j)}$ values. So, over the random choices of λ, the probability that the error term works out to be 0 anyways is equal to $1/p$. This allows us to establish the following Lemma:

Lemma 12. *If a witness (K, T) cannot be recovered from the queries to $\mathcal{H}_1$, then for any positive integer x, we have that for all query-response pairs to $\mathcal{H}_2$*

$$\Pr[\#G_2(h_1, \sigma_2, h_2) > x] leq \Pr[X + Y > x]$$

where X is a random variable distributed as in Lemma 11, and Y is distributed as $\max(Y_1, Y_2, \ldots, Y_{q_2})$ with each Y_i is i.i.d. $\mathcal{B}(M - X, \frac{1}{p})$.

This proof essentially states that we must either be in the first or second case, and that in the second case the probability that a round works out is bounded by $1/p$, as already discussed. Remaining details for this part of the proof can similarly be found in [12].

Cheating in the third phase. For round three, we continue with our characterization of the best possible input an adversary can construct. Recall that the input to $\mathcal{H}_3$ consists of an h_2 and the $\Delta z_e^{(k)}$ values. For an input-output pair, define the set $G_3(h_2, \sigma_3, h_3)$ as

- $\emptyset$ if h_2 is not the output of a previous query (h_1, σ_2) to $\mathcal{H}_2$, which in turn is associated with a query σ_1 that has already been made to $\mathcal{H}_1$. Otherwise, the signature will never verify.
- The values $e \in [M]$ such that the following procedure passes: trace back to the input to the fist phase and recover the seeds in order to generate the shares of K, T, $r^{(j)}$, $a^{(k)}$, $b^{(k)}$, $c^{(k)}$, and $z^{(k)}$. By summing over these shares and incorporating the committed to Δ values, we get the candidate state of the equations at the time when the $\epsilon^{(k)}$ values become defined. This in turn allows us to define the corresponding $\alpha^{(k)}$, $\beta^{(k)}$, $\gamma^{(k)}$, and $\omega^{(k)}$ values, by

$$\begin{aligned}
\alpha_e^{(1)} &= a_e^{(1)} + \epsilon_e^{(1)} \cdot T_e & \beta_e^{(1)} &= b_e^{(1)} + \sum_j \lambda_e^{(j)} r_e^{(j)} \\
\alpha_e^{(2)} &= a_e^{(2)} + \epsilon_e^{(2)} \cdot K_e & \beta_e^{(2)} &= b_e^{(2)} + z_e^{(1)} + \sum_j \lambda_e^{(j)} J_e^{(j)} r_e^{(j)} \\
\alpha_e^{(3)} &= a_e^{(3)} + \epsilon_e^{(3)} \cdot T_e & \beta_e^{(3)} &= b_e^{(3)} + \sum_j \lambda_e^{(j)} I_e^{(j)} r_e^{(j)} \\
\gamma_e^{(k)} &= \epsilon_e^{(k)} z_e^{(k)} - c_e^{(k)} + \alpha_e^{(k)} b_e^{(k)} + \beta_e^{(k)} a_e^{(k)} - \alpha_e^{(k)} \beta_e^{(k)} \\
\omega_e &= z_e^{(2)} + z_e^{(3)} + \sum_j \lambda_e^{(j)} I_e^{(j)} J_e^{(j)} r_e^{(j)} - \sum_j \lambda_e^{(j)} o_e^{(j)}.
\end{aligned}$$

  Then pass if all of $\gamma_e^{(k)}$ and $\omega_e^{(k)}$ are equal to zero.

Again, we need to bound the number of such rounds in the event that a witness cannot be recovered from the queries to $\mathcal{H}_1$. Clearly it must be the case that we can trace the query h_2 back to the original input to $\mathcal{H}_1$, or else the size of $G_3(h_2, \sigma_3, h_3)$ is zero. For each index $e \in [M]$, it may be the case that the index is in $G_2(h_1, \sigma_2, h_2)$. Assume it is not. Since the index is not in G_2, we can see that it must be the case that the error term is not equal to zero. A simple reduction shows that if $x^{(k)}$ and $y^{(k)}$ are the inputs to the kth multiplication gate, we have that,

$$\gamma_e^{(k)} = \epsilon_e^{(k)} cdot(z_e^{(k)} - x_e^{(k)} \cdot y_e^{(k)}) - c_e^{(k)} + a_e^{(k)} \cdot b_e^{(k)}.$$

We can see that this is equal to zero only if either:

– $z_e^{(k)} = x_e^{(k)} \cdot y_e^{(k)}$ and $c_e^{(k)} = a_e^{(k)} \cdot b_e^{(k)}$ (that is, the $\Delta z_e^{(k)}$ and $\Delta c_e^{(k)}$ values were chosen properly so that z and c really are equal to the product of the inputs).
– $\epsilon_e^{(k)} = (a_e^{(k)} \cdot b_e^{(k)} - c_e^{(k)})(z_e^{(k)} - x_e^{(k)} \cdot y_e^{(k)})^{-1}$.

As the $\epsilon^{(k)}$ values are chosen uniformly at random, for each of the γ values, the chances that $\gamma_e^{(k)} = 0$ *without* $z_e^{(k)} = x_e^{(k)} \cdot y_e^{(k)}$ is $1/p$. But if, for each k, we do have $z_e^{(k)} = x_e^{(k)} \cdot y_e^{(k)}$ then the corresponding ω_e value will in fact be equal to the error term. But since we are assuming that this round is not in $G_2(h_1, \sigma_2, h_2)$ we have that the error term is in fact not equal to zero. So we have that for least one of the $\gamma_e^{(k)}$ terms we must have that $z_e^{(k)}$ is not equal to the inputs. Thus the probability that all values are equal to zero is at most $3/p$.

This allows us, as in the previous round, to bound the size of $G_3(h_2, \sigma_3, h_3)$:

Lemma 13. *If a witness (K, T) cannot be recovered from the queries to $\mathcal{H}_1$, then for any positive integer x, we have that for all query-response pairs to $\mathcal{H}_3$*

$$\Pr[\#G_3(h_2, \sigma_3, h_3) > x] \leq \Pr[X + Y + Z > x]$$

where X is a random variable distributed as in Lemma 11, and Y is distributed as in 12, and Z is distributed as $\max(Z_1, Z_2, \ldots, Z_{q_2})$ with each Z_i is i.i.d. $\mathcal{B}(M - X - Y, \frac{3}{p})$.

Cheating in the fourth phase. Assume without loss of generality that $\mathcal{A}$ verifies the signature that they submit as a forgery. Recall that queries to $\mathcal{H}_4$ should take the form

$$h_3, \sigma_4 = \left((\alpha_e^{(k)}, \beta_e^{(k)}, (\alpha_{e,i}^{(k)}, \beta_{e,i}^{(k)}, \gamma_{e,i}^{(k)})_{i \in [N]})_{k \in [3]}, (\omega_{e,i})_{i \in [N]} \right)_{e \in [M]}.$$

For each such query σ_4 we can bound the probability that the response leads $\mathcal{A}$ to be able to construct a valid signature.

As usual, we note that h_3 must be the output of a query to $\mathcal{H}_3$ — if the adversary decided to come up with the h_3 value in some other way, then after they have queried it to $\mathcal{H}_4$ no input that they provide to $\mathcal{H}_3$ will lead to h_3. Similarly, we can work our way backwards to the original query to $\mathcal{H}_1$.

Assume that in a given round e we have that $e \notin G_3(h_2, \sigma_3, h_3)$. Then it is the case that at least one of the associated $\gamma_e^{(k)}$ or ω_e values is not equal to zero. In this case, the only way that we can end up with a valid signature is if exactly $N - 1$ of the parties behave honestly. If all parties behave honestly, then the final signature will not validate because the $\gamma_{e,\bar{i}_e}^{(k)}$ or $\omega_{e,\bar{i}_e}$ values will not be correct. If more than $N - 2$ parties misbehave then the verifier will not replicate that misbehaviour. With exactly $N - 1$ parties misbehaving, then for any round, the probability that that party is chosen to stay concealed by the protocol is clearly $1/N$. As a result, we can give our final bound on the probability that the adversary winning.

The reduction simulates all oracles correctly, except that it does not return collisions or allow the adversary to cheat at commitments to the random oracles.

In [12], the authors establish that the probability that an adversary notices this inconsistency is at most

$$\frac{MN(q_{sd}+q_1+q_2+q_3+q_4)^2}{2^{2\lambda}}. \tag{2}$$

And given that the adversary does not notice this incongruence we have that the probability that they are able to find a signature is less than

$$\Pr[X+Y+Z+W=M]$$

where X, Y, and Z are distributed according to Lemmas 11, 12, and 13, and W is distributed as $\max(W_1,\ldots,W_{q_4})$ with each W_i distributed i.i.d. as $\mathcal{B}(M-X-Y-Z,1/N)$.

EU-CMA implies key-only security and signing with oracle reprogramming In order to simulate signatures with oracle reprogramming, we can define our Forge function as follow.

1. To cheat in the first round, the simulator simply samples the values ΔK_e and ΔT_e uniformly at random, rather than calculating them using T. The simulator also aborts if a salt is reused from an earlier round. The reduction queries the random oracle (which is also managed by them) to obtain h_1, which is expanded to the $I_e^{(j)}, J_e^{(j)}$ values.
2. For the second part of proving, rather than genuinely generating the $o_e^{(j)}$ values, the simulator will sample them uniformly and then post select so that $\mathcal{L}^\ell(o_e^{(j)}) - s_e^{(j)} = blpk_{I_e^{(j)},J_e^{(j)}}$. Then they again query this to the random oracle to obtain the $\lambda_e^{(j)}$ values.
3. For prover part 3, simply select each $\Delta z_e^{(k)}$ value to be uniformly random. Set σ_3 in the usual way and query for the $\epsilon_e^{(k)}$ values.
4. For the simulation of prover part 4, the simulator will need to cheat so the openings can be provided for the requested parties that look correct, and the overall proof lines up. To do this, they will decide which parties will be opened in advance, and then later program the random oracle to provide this output. So for each $e \in [M]$, the simulator samples $\bar{i}_e \in \{1,\ldots,N\}$ at random. Then it behaves entirely honestly except for when working with the values for the $\bar{i}_e$th party in round e. This party is calculated last, and we set the γ and ω values according to how they are set in verification, rather than how they are set in an honest signing instance. That is, we set

$$\gamma_{e,\bar{i}_e}^{(k)} \leftarrow -\sum_{i\neq\bar{i}_e} \gamma_{e,i}^{(k)} \qquad\qquad \omega_{e,\bar{i}_e} \leftarrow -\sum_{i\neq\bar{i}_e} \omega_{e,i}.$$

   Then σ_4 is prepared accordingly. Rather than querying the random oracle $\mathcal{H}_4$, we instead reprogram it so that (h_3,σ_4) maps to the desired $(\bar{i}_e)_{e\in[M]}$ values.
5. Finally for part 5, the simulator can generate the seeds as desired and return the signature as expected.

Simulation Indistinguishability. The random oracles are all simulated perfectly. The programmed output is chosen uniformly at random, and so as long as it has not been previously queried, the programmed output is indistinguishable from a real one. As the oracle $\mathcal{H}_4$ is reprogrammed on the point (h_3, σ_4), this is the output of the Ext function. That is, Ext takes in the signature σ, uses it to reconstruct h_3 and σ_4 according to the Verify function, and returns them as the reprogrammed point.

The min-entropy of this point is quite high. We need to consider the min-entropy of this input from the beginning of the signing routine. All of the α, β, and γ values are derived in part from the a, b, c, and z values. These in turn are derived be expanding the seeds, which are chosen from the uniformly sampled root seed sd_e. Therefore the entropy of the programmed point is bounded by the entropy of the sampled root vector. As a root vector is sampled for each round, the min entropy of the overall programmed point can be bounded by $2^{\lambda \cdot M}$.

We must consider the distribution of the reprogrammed point and the returned signature. Most of the parts of the signature are exactly the same as what they would have been without any cheating (just possibly calculated a different way), except for the Δ correction values: ΔK_e, ΔT_e, and $\Delta z_e^{(k)}$ are all chosen uniformly at random rather than calculated based on K, T, and the output values. But, this does not actually change the overall distribution of the signatures. Since the unknown shares $K_{e,\bar{i}_e}$, $T_{e,\bar{i}_e}$, $z_{e,\bar{i}_e}^{(k)}$ are never seen by verifier, and are generated uniformly from $\mathsf{Expand}(\mathsf{sd}_{e,\bar{i}_e})$, they just as easily could have been the values such that the Δ values are calculated correctly. Therefore the distribution, conditioned on what a verifier sees as part of the signature, is exactly the same, and we can sign with oracle reprogramming.

Similarly, the distribution of the programmed output, y_{forged} is entirely unchanged from y_{real}, as both are taken uniformly from $\{1, \ldots, N\}^M$, and independent of anything else in the signatures. So we can conclude that the statistical distance δ between our real and forged σ and y is in fact 0.

For the complete details on how signing with oracle reprogramming shows that an adversary attacking the existential unforgeability can be used to construct an adversary attacking the key-only security, we refer to the LegRoast paper [12], which shows exactly this in their Lemma 3.