Fusion One-Time Non-Interactively-Aggregatable Digital Signatures From Lattices

Brandon Goodell[1] and Aaron Feickert[2]

[1] Geometry Labs
brandon@geometrylabs.io
[2] Cypher Stack
aaron@cypherstack.com

Abstract. We present Fusion, a post-quantum one-time digital signature scheme with non-interactive aggregation with security resting on the short integer solution problem over ideal lattices. Fusion is structurally similar to CRYSTALS-Dilithium, but Fusion is based upon the aggregatable one-time lattice-based scheme by Boneh and Kim. Fusion parameters conservatively target at least 128 bits of security against forgery, taking tightness gaps into account, and with tighter bounds than the BK scheme. Aggregate Fusion signatures are logarithmically sized in the number of keys, so aggregating enough signatures can be more efficient than stacking Dilithium or Falcon signatures.

Acknowledgements. Funding for this work was provided by the Quantum Resistant Ledger Foundation.

1 Advantages and Limitations

Fusion signatures enjoy the following advantages.

- Unlike predecessors, Fusion signatures are aggregatable. Aggregate Fusion signatures are logarithmically sized in the number of keys, and capacities can exceed 32000. Aggregating enough signatures can be more efficient than stacking Dilithium or Falcon signatures.
- Fusion is conservatively parameterized, taking tightness gaps into account, enjoys tighter bounds than [1], and has smaller parameters, keys, and signatures than the estimates of [1]. Fusion parameterization is modular so only a few parameters need to be changed in order to change security.
- Fusion enjoys structural similarity to CRYSTALS-Dilithium, a heavily vetted scheme selected by NIST, and so benefits from a large community of informed well-equipped reviewers.
- Fusion implementation is simple: one-time signatures ξ are weighted and bounded linear combinations $\xi = \underline{f}_0 c + \underline{f}_1$ of secret vectors of polynomials $\underline{f}_0, \underline{f}_1$, and aggregate one-time signatures ξ_{ag} are weighted and bounded linear combinations $\sum_i \alpha_i \xi_i$ of signatures.
- Fusion arithmetic is fast thanks to the number theoretic transform (**NTT**).
- Fusion signatures are one-time, sidestepping many side-channel attacks and the problems of statefulness, randomization, and aborts.
- Fusion avoids trapdoor sampling, Gaussian sampling, and the additional structure imposed by the NTRU equations used in Falcon.

Fusion signatures have the following limitations.

- Aggregating Fusion one-time signatures can be more efficient than stacking CRYSTALS-Dilithium and Falcon signatures naively, but there is a critical number of one-time signatures needed for that performance. Users only interested in verifying a small number of signatures at a time can do better by using CRYSTALS-Dilithium or Falcon.
- Un-aggregated Fusion one-time signatures must be passed around to an aggregator, introducing a communication overhead.
- Signing, aggregation, and verification are much faster thanks to the **NTT**. However, the **NTT** introduces a tradeoff between timing and space efficiency which may be unsuitable for some use-cases.
- As a lattice-based scheme similar to CRYSTALS-Dilithium, Fusion benefits from an experience community of reviewers, but does not contribute to diversity of hardness assumptions underlying NIST algorithms.

1.1 Performance and Parameterization

We present three parameterizations of Fusion signatures named by key sizes: Light, Mid, and Heavy. Fusion Light is tuned to have 128 bits of security against forgery. For Fusion Mid and Fusion Heavy, we have two sub-parameterizations with 128 and 256 bits of security.

Comparing the space complexity of Fusion with CRYSTALS-Dilithium or Falcon requires taking into account that Fusion keys are one-time keys, whereas Dilithium and Falcon keys are many-time keys. Indeed, a single Dilithium or Falcon key may be posted and many signatures on many messages generated from that one key. The average space complexity per message of using Dilithium or Falcon asymptotically approaches the cost of signatures alone. On the other hand, the space complexity of Fusion one-time aggregate signatures includes all keys and the aggregate signature. In this way, it is only fair to compare both keys and the aggregate signature of Fusion against the signatures alone in Dilithium or Falcon. Moreover, in this way, it is easy to compute how many signatures must be aggregated for performance to beat Dilithium or Falcon.

- Fusion Light has 128 bits of security. Keys are 496 bytes, signatures are 21.84 kilobytes, and we can combine up to 1796 signatures into an aggregate signature with 46.8 kilobytes at 128 bits of security. Aggregating at least 25 signatures is more space-efficient than CRYSTALS-Dilithium, and aggregating at least 276 signatures is more space-efficient than Falcon, but otherwise naively stacking Dilithium or Falcon signatures is more space efficient.
- Fusion Mid has 128 or 256 bits of security. Keys are 992 bytes in both cases. Signatures are 17.072 and 42.496 kilobytes, respectively. We can combine 20813 or 236 signatures, respectively, into an aggregate signature with 46.56 or 79.68 kilobytes, respectively. Aggregating at least 33 or 56 signatures, respectively, is more space-efficient than CRYSTALS-Dilithium, but this parameterization is not more efficient than Falcon.
- Fusion Heavy has 128 or 256 bits of security. Keys are 1984 bytes in both cases. Signatures are 16.896 or 34.528 kilobytes, respectively. We can combine 32417 or 2818 signatures, respectively, into an aggregate signature with 46.08 or 79.68 kilobytes, respectively. Aggregating at least 106 or 183 signatures, respectively, is more space-efficient than CRYSTALS-Dilithium, but this parameterization is not more efficient than Falcon.

1.2 Bird's Eye View

The following is a high level summary of Fusion signatures. We include brief descriptions of notation and operators here, but we elaborate on these in section 2. Let d be a power of two, p a prime, $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$, $\mathfrak{p}$ the ideal in $\mathcal{R}$ principally generated by p, $\mathcal{R}_\mathfrak{p} = \mathcal{R}/\mathfrak{p}\mathcal{R}$

the quotient ring, ℓ a natural number, μ the Hamming weight on $\mathcal{R}_\mathfrak{p}$, $\|\cdot\|_\infty$ the usual infinity norm on $\mathcal{R}_\mathfrak{p}$, and let $K, \beta_{sk}, \omega_{sk}, \beta_{ch}, \omega_{ch}, \beta_{ag}, \omega_{ag}, \beta_v, \omega_v \in \mathbb{N}$.

1. During a setup phase, a public $\underline{a} \leftarrow\$\, \mathcal{R}_\mathfrak{p}^\ell$ is sampled uniformly.
2. A secret one-time signing key is $\mathsf{sk} = (\underline{f}_0, \underline{f}_1) \in \mathcal{R}_\mathfrak{p}^\ell \times \mathcal{R}_\mathfrak{p}^\ell$ sampled from a distribution that is statistically close to uniform such that $\left\|\underline{f}_i\right\|_\infty \leq \beta_{sk}$ and $\mu(\underline{f}_i) = \omega_{sk}$ for each $i = 0, 1$. The corresponding one-time public verification key is $\mathsf{vk} = (g_0, g_1) = (\langle \underline{a}, \underline{f}_0 \rangle, \langle \underline{a}, \underline{f}_1 \rangle)$.
3. The one-time signature for $m \in \{0,1\}^*$ is $\underline{\xi} = \underline{f}_0 c + \underline{f}_1$ where the one-time signature challenge is $c = \mathsf{H}_{ch}(\mathsf{vk}, m)$ for a hash function H_{ch} such that $\|c\|_\infty \leq \beta_{ch}$ and $\mu(c) = \omega_{ch}$.
4. An aggregate one-time signature is $\underline{\xi}_{ag} = \sum_{i=0}^{N-1} \alpha_i \underline{\xi}_i$ for some $1 \leq N \leq K$ where each $\underline{\xi}_i$ is a one-time signature corresponding to a message m_i, public verification key vk_i, and signature challenge c_i, and where the aggregation coefficients are $(\alpha_0, \ldots, \alpha_{N-1}) = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i=0}^{N-1})$ for a hash function H_{ag} such that $\|\alpha_i\|_\infty \leq \beta_{ag}$ and $\mu(\alpha_i) = \omega_{ag}$ for each $i \in [K]$.
5. A aggregate one-time signature $\underline{\xi}_{ag}$ for messages $m_0, m_1, \ldots, m_{N-1}$, sorted public verification keys $\mathsf{vk}_0, \mathsf{vk}_1, \ldots, \mathsf{vk}_{N-1}$, signature challenges $c_0, c_1, \ldots, c_{N-1}$, and aggregation coefficients $(\alpha_0, \alpha_1, \ldots, \alpha_{N-1})$ is valid if all of the following hold.
   (i) each $c_i = \mathsf{H}_{ch}(\mathsf{vk}_i, m_i)$,
   (ii) $(\alpha_0, \ldots, \alpha_{N-1}) = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i=0}^{N-1})$,
   (iii) $\left\|\underline{\xi}_{ag}\right\|_\infty \leq \beta_v$,
   (iv) $\mu(\underline{\xi}_{ag}) \leq \omega_v$, and
   (v) $\langle \underline{a}, \underline{\xi}_{ag} \rangle = \sum_{i \in [N]} \alpha_i (g_{i,0} c_i + g_{i,1})$ for $\underline{\alpha} = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i \in [N]})$.

Revealing two distinct signatures on the same keys provides enough information to extract the keys, so this scheme is one-time only. Sorting before verification guarantees that the order of the input keys does not impact the aggregate signature.

In practice, $\underline{a}$ can be expanded using an XOF from a public seed, say σ, which can be posted with keys and aggregate signatures, and which can serve as a domain-separating salt for hash functions H_{ch} and H_{ag}. Also, sk can be expanded using an XOF from a secret seed, say η, perhaps a hash of some public document, or the output from some KEM. These choices can help to ensure that an adversary would have to control the output of the XOF in order to manipulate $\underline{a}$ or sample keys maliciously.

The number-theoretic transform (**NTT**) speeds up arithmetic at no security cost, but generally increases space complexity. Thus, in practice, it also may be beneficial to post the transforms $(\widehat{g}_0, \widehat{g}_1) = (\mathbf{NTT}(g_0), \mathbf{NTT}(g_1))$, $\widehat{\underline{\xi}} = \mathbf{NTT}(\xi)$, and $\widehat{\underline{\xi}}_{ag} = \mathbf{NTT}(\xi_{ag})$ instead of (g_0, g_1), ξ, or ξ_{ag}, respectively. Note, however, that some transforms are unavoidable. Indeed, verifiers need to know both $\widehat{\xi}_{ag}$ and ξ_{ag} to compute the norm and the inner product transforms during verification.

1.3 Improvements over [1]

A version of this scheme was first described in [1] by Boneh and Kim, and it is based on the scheme described in [2] by Lyubashevsky and Micciancio. The differences between Fusion signatures and the scheme presented by [1] are small but have important impacts on our work. Our proof of Theorem 6 in particular improves that of [1] with a smaller extracted witness and without a reduction to a selective forger.

- We include signature challenges in the computation of the aggregation coefficients: we compute $\underline{\alpha} = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i \in [N]})$ instead of $\underline{\alpha} = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i)_{i \in [N]})$. We suspect that we could even use $\underline{\alpha} = \mathsf{H}_{ag}((\mathsf{vk}_i, c_i)_{i \in [N]})$ instead and retain security. By including signature challenges in the computation of aggregation coefficients, we can remove the selective forgery approach and present a moderately tighter unforgeability proof than the approach in [1].

- We specify Hamming weight bounds for keys, signature challenges, and aggregation coefficients, ω_{sk}, ω_{ch}, ω_{ag} and carefully apply the properties of $\|\cdot\|$ and $\mu(\cdot)$ to derive bounds in all proofs. By restricting the Hamming weights of keys, signature challenges, and aggregation coefficients, we also obtain a yet-tighter proof than the one in [1]. As we shall see, we end up with sparse signature challenges and aggregation coefficients with $\omega_{ch} < d$ and $\omega_{ag} < d$, but we also end up with dense keys with $\omega_{sk} = d$. We include ω_{sk} throughout our analysis for clarity and completeness.

2 Preliminary Information

2.1 General Preliminaries

All logarithms in the sequel are base 2. We denote the integers with $\mathbb{Z}$, we denote the natural numbers with $\mathbb{N}$, and the non-negative integers with $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For $k \in \mathbb{N}$, denote the subset $\{0, 1, \ldots, k-1\} \subseteq \mathbb{N}_0$ with $[k]$. We generally use underlines for lists/vectors/module elements, which we index with $\mathbb{N}_0$. For example, if $\mathcal{R}$ is a ring, $\ell \in \mathbb{N}$, and $\mathcal{V} = \mathcal{R}^\ell$ denotes the rank-ℓ free $\mathcal{R}$-module, we denote an element of $\mathcal{V}$ with $\underline{f} = (f_0, \ldots, f_{\ell-1})$.
For parameter sets $\Lambda_0, \Lambda_1, \ldots, \Lambda_{T-1}$, $\mathcal{P} = \Lambda_0 \times \Lambda_1 \times \cdots \Lambda_{T-1}$, and for $\underline{\lambda} = (\lambda_0, \lambda_1, \ldots, \lambda_{T-1}) \in \mathcal{P}$, we use $\mathsf{poly}(\underline{\lambda})$ to denote the class of all functions $f : \mathcal{P} \to \mathbb{N}$ such that there exists some polynomial $p(\underline{\lambda})$ such that $f \in O(p)$. We call all such f *polynomially bounded.* For a parameter set Λ, we use $\mathsf{negl}(\lambda)$ to denote the class of functions $f : \Lambda \to \mathbb{N}$ such that $f = O(1/p)$ for every $p \in \mathsf{poly}(\lambda)$. We call all such f *negligible in* λ.
For an event E, we denote the probability of E by $\mathbb{P}[E]$. For any set X, we write $x \leftarrow\!\!\$\ X$ to indicate that we have sampled $x \in X$ uniformly at random, and independent of all other random variables. For a randomized algorithm $\mathcal{A}$, we write $a \leftarrow \mathcal{A}$ to indicate that a is a random output from $\mathcal{A}$.

2.2 One-Time Aggregatable Signatures and Security Properties

We now define one-time aggregatable signature schemes, their correctness, and their one-time unforgeability against chosen message attacks.

Definition 1. *An aggregatable signature scheme is a tuple of the following algorithms.*

1. $\mathsf{Pgen}(\lambda) \to \rho$. *Input a security parameter* $\lambda \in \mathbb{N}$*, and output some public parameters* ρ.
2. $\mathsf{KGen}(\rho) \to (\mathsf{sk}, \mathsf{vk})$. *Input* ρ *and output a new secret-public keypair* $(\mathsf{sk}, \mathsf{vk})$.
3. $\mathsf{Sign}(\rho, \mathsf{sk}, m) \to \xi$. *Input* ρ*, a secret key* sk*, and a message* m*. Output a signature* ξ.
4. $\mathsf{Agg}(\rho, (\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]}) \to \xi_{ag}$. *Input* ρ *and a list of public verification key-message-signature triples* $(\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]}$*. Output an aggregated signature* ξ_{ag}.
5. $\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i \in [N]}, \xi_{ag}) \to b \in \{0, 1\}$. *Input* ρ*, a list of public verification key-message pairs* $(\mathsf{vk}_i, m_i)_{i \in [N]}$*, and an aggregate signature* ξ_{ag}*. Output a bit.*

In the sequel, we let $\Pi = (\mathsf{Pgen}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Agg}, \mathsf{AggVf})$ denote an aggregatable signature scheme, and for some $\lambda \in \mathbb{N}$, we let $E(\Pi, \lambda)$ denote the event in which ξ_{ag} is a semi-honestly computed aggregate signature in the sense that $\rho \leftarrow \mathsf{Pgen}(\lambda)$, $K \in \rho$, $1 \leq N \leq K$, $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(\rho)$ for each $i \in [N]$, $\xi_i \leftarrow \mathsf{Sign}(\rho, (\mathsf{sk}_i, \mathsf{vk}_i), m_i)$ for each $i \in [N]$, and $\xi_{ag} \leftarrow \mathsf{Agg}(\rho, (\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]})$.

Definition 2. *If* $1 = \mathbb{P}\left[\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i \in [N]}, \xi_{ag}) = 1 \mid E(\Pi, \lambda)\right]$ *then we say* Π *is a correct aggregatable signature scheme.*

Definition 3. *If Π is correct and $1 = \mathbb{P}\left[|\xi_{ag}| \in \mathsf{poly}(\lambda, \log(K)) \mid E(\Pi, \lambda)\right]$ where $|\xi_{ag}|$ denotes the space complexity of ξ_{ag}, then we say Π is compact.*

Definition 4. *Let $\mathcal{A}$ be any PPT algorithm and assume Π is correct. Let $E'(\Pi, \lambda)$ be the event in which (i) $1 \leq N \leq K$, (ii) there exists some $i^* \in [N]$ such that $\mathsf{vk}_{i^*} = \mathsf{vk}^*$, $m_{i^*} = m_{\mathcal{O}}$, and $\xi_{i^*} = \xi_{\mathcal{O}}$, (iii) $(\mathsf{sk}_i, \mathsf{vk}_i) \leftarrow \mathsf{KGen}(\rho)$ for each $i^* \neq i \in [N-1]$, (iv) each m_i is a bit message, (v) $\xi_i \leftarrow \mathsf{Sign}(\rho, (\mathsf{sk}_i, \mathsf{vk}_i), m_i)$ for each $i^* \neq i \in [N]$, and (vi) the list of tuples $(\mathsf{vk}_i, m_i, \xi_i)_{i\in[N]}$ have been sorted by the keys vk_i, and (vii) $\xi_{ag} \leftarrow \mathsf{Agg}(\rho, (\mathsf{vk}_i, m_i, \xi_i)_{i\in[N]})$. The following game defines one-time existential unforgeability against chosen message attack.*

1. *The challenger and the player agree upon λ. The challenger computes $\rho \leftarrow \mathsf{Pgen}(\lambda)$ and then samples $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{KGen}$. The challenger sends (ρ, vk^*) to $\mathcal{A}$.*
2. *The challenger grants $\mathcal{A}$ one-time signing oracle access to get one signature $\xi_{\mathcal{O}}$ on an oracle query message $m_{\mathcal{O}}$ at any time. The oracle query response $\xi_{\mathcal{O}}$ is such that*

$$\mathbb{P}[\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i\in[N]}, \xi_{ag}) = 1 \mid E'(\Pi, \lambda)] = 1.$$

3. *The player $\mathcal{A}$ eventually outputs a forgery $\mathsf{forg} = ((\mathsf{vk}_i, m_i)_{i\in[N]}, \xi_{ag})$ where each vk_i is a public verification key, each m_i is a bit message, and ξ_{ag} is a purported aggregate signature.*

The oracle response satisfies $\mathbb{P}[\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i\in[N]}, \xi_{ag}) = 1 \mid E'(\Pi, \lambda)] = 1$, guaranteeing that $\xi_{\mathcal{O}}$ is aggregatable with up to $K-1$ semi-honestly computed signatures on any messages using semi-honestly generated verification keys (i.e. in event $E'(\Pi, \lambda)$). The player $\mathcal{A}$ succeeds if and only if there exists an index $i^* \in [N]$ such that $\mathsf{vk}_{i^*} = \mathsf{vk}^*$, the oracle was not queried with the corresponding message $m_{i^*} \neq m_{\mathcal{O}}$, and $\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i\in[N]}, \xi_{ag}) = 1$. If $\mathcal{A}$ runs in time t and succeeds at this game with probability at least ϵ, we say $\mathcal{A}$ is a (t, ϵ)-forger. If every PPT algorithm that is a (t, ϵ)-forger for some $t \in \mathsf{poly}(\lambda)$ also has $\epsilon \in \mathsf{negl}(\lambda)$, then we say Π is one-time existentially unforgeable.

2.3 Rings, norms, and weights

Let X be indeterminate over $\mathbb{Z}$, and let $\mathbb{Z}[X]$ be the set of polynomials with coefficients from $\mathbb{Z}$. Let $e, p, \ell \in \mathsf{poly}(\lambda)$ such that $e(\lambda), p(\lambda), \ell(\lambda) \in \mathbb{N}$ for every λ and each $p(\lambda)$ is prime. Set $d(\lambda) = 2^{e(\lambda)}$, $\mathcal{R}(\lambda) = \mathbb{Z}\left[X\right]/(X^{d(\lambda)}+1)$, and $\mathcal{V}(\lambda) = \mathcal{R}(\lambda)^{\ell(\lambda)}$. Here, e is for exponent, p is for prime, ℓ is for length, d is for degree, $\mathcal{R}$ is for ring, and $\mathcal{V}$ is for vectors (abusing terminology since $\mathcal{V}$ is an $\mathcal{R}$-module). Let $\mathfrak{p}(\lambda)$ denote the principal ideal in $\mathbb{Z}$ generated by $p(\lambda)$. An element $f \in \mathcal{R}(\lambda)$ is a coset of polynomials from $\mathbb{Z}[X]$ all equivalent modulo $X^{d(\lambda)} + 1$. Due to the polynomial modulus, without loss of generality, we represent f with a polynomial from $\mathbb{Z}[X]$ with degree bound $d(\lambda)$, say $f(X) = \sum_{i\in[d(\lambda)]} f_i X^i$. An element $\underline{f} \in \mathcal{V}(\lambda)$ is an $\ell(\lambda)$-tuple of polynomials from $\mathcal{R}(\lambda)$. Since everything depends on λ, we drop λ from our notation in the sequel for clarity, and we caution readers to keep the dependence on λ in mind.

Note $\mathfrak{p}$ is a principal ideal in $\mathcal{R}(\lambda)$ via the canonical inclusion mapping $\mathbb{Z} \hookrightarrow \mathcal{R}$. Let $\mathbb{Z}_{\mathfrak{p}} = \mathbb{Z}/\mathfrak{p}\mathbb{Z}$, $\mathcal{R}_{\mathfrak{p}} = \mathcal{R}/\mathfrak{p}\mathcal{R}$, and $\mathcal{V}_{\mathfrak{p}} = \mathcal{R}_{\mathfrak{p}}^{\ell}$. An element of $\mathbb{Z}_{\mathfrak{p}}$ is a coset of integers modulo p. An element of $\mathcal{R}_{\mathfrak{p}}$ is a coset of polynomials from $\mathcal{R}$ modulo p. For any $f \in \mathcal{R}_{\mathfrak{p}}$, we assume without loss of generality that the coefficients have representatives in $\left\{0, \pm 1, \pm 2, \ldots, \pm \frac{p-1}{2}\right\}$. Since $\mathcal{V}_{\mathfrak{p}} = \mathcal{R}_{\mathfrak{p}}^{\ell}$ for some ℓ, an element of $\mathcal{V}_{\mathfrak{p}}$ is an ℓ-tuple of elements from $\mathcal{R}_{\mathfrak{p}}$. $\mathcal{V}$ is an $\mathcal{R}$-module and $\mathcal{V}_{\mathfrak{p}}$ is an $\mathcal{R}_{\mathfrak{p}}$-module, but $\mathcal{V}_{\mathfrak{p}}$ is also a vector space over $\mathbb{Z}_{\mathfrak{p}}$, which justifies our use of $\mathcal{V}$ for vectors. The ring $\mathcal{R}$ admits the Hamming weight function $\mu : \mathcal{R} \to \mathbb{N}$ by counting the non-zero coefficients of $f \in \mathcal{R}$, i.e. $f \mapsto \sum_{i\in[d]} \mathbb{1}(f_i \neq 0) \in \mathbb{N}_0$. The Hamming weight function μ extends naturally to $\mathcal{R}_{\mathfrak{p}}$ and, by defining $\mu(\underline{f}) = \max_{j\in[\ell]} \mu(\underline{f}_j)$ on $\mathcal{V}$ and $\mathcal{V}_{\mathfrak{p}}$, the Hamming weight extends to $\mathcal{V}$ and $\mathcal{V}_{\mathfrak{p}}$, also. The function μ enjoys the following properties.

Property 1. For any $(f, g, \underline{h}, \underline{h}') \in \mathcal{R} \times \mathcal{R} \times \mathcal{V} \times \mathcal{V}$ or in $\mathcal{R}_{\mathfrak{p}} \times \mathcal{R}_{\mathfrak{p}} \times \mathcal{V}_{\mathfrak{p}} \times \mathcal{V}_{\mathfrak{p}}$, we have
(a) $\mu(f+g) \leq \min(d, \mu(f) + \mu(g))$,
(b) $\mu(fg) \leq \min(d, \mu(f)\mu(g))$,
(c) $\mu(\underline{h} + \underline{h}') \leq \min(d, \mu(\underline{h}) + \mu(\underline{h}'))$, and
(d) $\mu(f\underline{h}) \leq \min(d, \mu(f)\mu(\underline{h}))$.

Proof. For (a), note $\mu(f+g)$ is the number of non-zero coefficients of $f+g$, and the i^{th} coefficient of $f+g$ is exactly $f_i + g_i$ where f_i is the i^{th} coefficient of f and g_i is the i^{th} coefficient of g. At most $\mu(f) + \mu(g)$ of these can be non-zero, and any element of $\mathcal{R}$ has at most d coefficients. For (b), $\mu(fg)$ is the number of non-zero coefficients of fg. Write f as a sum of $\mu(f)$ terms, namely $f = \sum_{i \in [\mu(f)]} f_{\phi(i)} X^{\phi(i)}$ for an injective function $\phi : [\mu(f)] \to [d]$, and write $g = \sum_{i \in [\mu(g)]} g_{\psi(i)} X^{\psi(i)}$ for an injective $\psi : [\mu(g)] \to [d]$. Using these, $fg = \sum_{i \in [\mu(f)]} \sum_{i' \in [\mu(g)]} f_{\phi(i)} g_{\psi(i')} X^{\phi(i)+\psi(i')}$. Thus, we have at most $\mu(f)\mu(g)$ non-zero monomials in this sum (and generally fewer). Any sum of monomials in $\mathcal{R}$ is a polynomial with at most d non-zero coefficients. For (d), apply (b) to each coordinate of $f\underline{h}$, and for (c), apply (a) to each coordinate of $\underline{h} + \underline{h}'$.

The ring $\mathcal{R}$ admits the functions $\|\cdot\|_\infty, \|\cdot\|_2 : \mathcal{R} \to \mathbb{N}$ by mapping $f \mapsto \max_{i \in [d]} |f_i|$ and $f \mapsto \left(\sum_i f_i^2\right)^{1/2}$, respectively. These both extend to $\mathcal{V}$ in the usual way: $\left\|\underline{f}\right\|_\infty = \max_{j \in [\ell], i \in [d]} |\underline{f}_{j,i}|$ and $\left\|\underline{f}\right\|_2^2 = \sum_{j \in [\ell]} \left\|\underline{f}_j\right\|_2^2$. These functions are not norms, but they enjoy the following properties.

Property 2. For any $(f, g, \underline{h}, \underline{h}') \in \mathcal{R} \times \mathcal{R} \times \mathcal{V} \times \mathcal{V}$, we have
(a) $\|f+g\|_\infty \leq \|f\|_\infty + \|g\|_\infty$ (triangle inequality),
(b) $\|\underline{h} + \underline{h}'\|_\infty \leq \|\underline{h}\|_\infty + \|\underline{h}'\|_\infty$ (triangle inequality),
(c) $\|fg\|_\infty \leq \min(d, \mu(f), \mu(g))\|f\|_\infty \|g\|_\infty$,
(d) $\|f\underline{h}\|_\infty \leq \min(d, \mu(f), \mu(\underline{h}))\|f\|_\infty \|\underline{h}\|_\infty$,
(e) $\|f\|_\infty \leq \|f\|_2 \leq \sqrt{\mu(f)}\|f\|_\infty$, and
(f) $\|\underline{h}\|_\infty \leq \|\underline{h}\|_2 \leq \sqrt{\ell\mu(\underline{h})}\|\underline{h}\|_\infty$.

Proof. Note that (a) is due to the fact that $\max_i |f_i + g_i| \leq \max_i |f|_i + \max_i |g|_i$. To obtain (b), apply (a) to each coordinate. For (c), write $f = \sum_{i \in [\mu(f)]} f_{\phi(i)} X^{\phi(i)}$ and $g = \sum_{i \in [\mu(g)]} g_{\psi(i)} X^{\psi(i)}$ for some injective functions ϕ, ψ as we did above.

$$\begin{aligned}
\|fg\|_\infty =& \left\| \sum_{i \in [\mu(f)]} f_{\phi(i)} X^{\phi(i)} g \right\|_\infty \leq \sum_{i \in [\mu(f)]} \left\| f_{\phi(i)} X^{\phi(i)} g \right\|_\infty \\
=& \sum_{i \in [\mu(f)]} \left\| f_{\phi(i)} g \right\|_\infty = \sum_{i \in [\mu(f)]} \max_{i' \in [\mu(g)]} |f_{\phi(i)} g_{\psi(i')}| \\
=& \sum_{i \in [\mu(f)]} |f_{\phi(i)}| \max_{i' \in [\mu(g)]} |g_{\psi(i')}| = \sum_{i \in [\mu(f)]} |f_{\phi(i)}| \|g\|_\infty \\
=& \|g\|_\infty \sum_{i \in [\mu(f)]} |f_{\phi(i)}| \leq \|g\|_\infty \sum_{i \in [\mu(f)]} \|f\|_\infty \\
=& \mu(f) \|f\|_\infty \|g\|_\infty
\end{aligned}$$

But $fg = gf$, so the conclusion is symmetric in f and g. Moreover, $\mu(f), \mu(g) \leq d$, so $\|fg\|_\infty \leq \min(d, \mu(f), \mu(g))\|f\|_\infty \|g\|_\infty$. To obtain (d), apply (c) to each coordinate.

Now consider (e). Note that $\max_i f_i^2 = \max_i |f_i|^2 = (\max_i |f_i|)^2 = \|f\|_\infty^2$, and this f_i^2 is included in the terms among the sum of positive terms $\|f\|_2^2 = \sum_i f_i^2$. Hence, $\|f\|_\infty^2 \leq \|f\|_2^2$. On the other hand, $\|f\|_2^2 = \sum_i f_i^2$. Recalling $f = \sum_{i \in [\mu(f)]} f_{\phi(i)} X^{\phi(i)}$, this sum has at most $\mu(f)$ non-zero terms. For each of these non-zero terms, $f_i^2 \leq \|f\|_\infty^2$. Hence, $\|f\|_2^2 \leq \mu(f)\|f\|_\infty^2$.

To obtain (f), we just use (e). Consider the lower bound. Note that $\|\underline{h}\|_\infty^2 = (\max_{i \in [\ell]} \|h_i\|_\infty)^2 = \max_{i \in [\ell]} \|h_i\|_\infty^2$. By (e), each $\|h_i\|_\infty^2 \leq \|h_i\|_2^2$. So $\max_{i \in [\ell]} \|h_i\|_\infty^2 \leq \max_{i \in [\ell]} \|h_i\|_2^2$ and $\|\underline{h}\|_\infty^2 \leq \max_{i \in [\ell]} \|h_i\|_2^2$. Note that this term $\max_{i \in [\ell]} \|h_i\|_2^2$ is one of the positive terms in the sum $\sum_i \|h_i\|_2^2 = \|\underline{h}\|_2^2$, so $\max_{i \in [\ell]} \|h_i\|_2^2 \leq \|\underline{h}\|_2^2$. Therefore, $\|\underline{h}\|_\infty^2 \leq \|\underline{h}\|_2^2$.

Next consider the upper bound and look at $\|\underline{h}\|_2^2 = \sum_{i \in [\ell]} \|h_i\|_2^2$. By (e), $\|h_i\|_2^2 \leq \mu(h_i)\|h_i\|_\infty^2$. Hence, $\|\underline{h}\|_2^2 \leq \sum_{i \in [\ell]} \mu(h_i)\|h_i\|_\infty^2$. But $\mu(\underline{h}) = \max_i \mu(h_i)$, so $\mu(\underline{h} \geq \mu(h_i)$ for each i. Moreover, $\|h_i\|_\infty^2 \leq \max_i \|h_i\|_\infty^2 = \|\underline{h}\|_\infty^2$. Thus, $\|\underline{h}\|_2^2 \leq \ell\mu(\underline{h})\|\underline{h}\|_{\infty^2}$.

These properties extend to $\mathcal{R}_\mathfrak{p}$ and $\mathcal{V}_\mathfrak{p}$ mostly as expected, with a caveat for comparing $\|\cdot\|_2$ and $\|\cdot\|_\infty$ to prevent wrap-around due to the prime modulus.

Property 3. For any $f, g \in \mathcal{R}_\mathfrak{p}$, for any $\underline{h} \in \mathcal{V}_\mathfrak{p}$, we have all of the following.
(a) $\|f + g\|_\infty \leq \|f\|_\infty + \|g\|_\infty$ (triangle inequality),
(b) $\|\underline{h} + \underline{h}'\|_\infty \leq \|\underline{h}\|_\infty + \|\underline{h}'\|_\infty$ (triangle inequality),
(c) $\|fg\|_\infty \leq \min(d, \mu(f), \mu(g))\|f\|_\infty \|g\|_\infty$,
(d) $\|f\underline{h}\|_\infty \leq \min(d, \mu(f), \mu(\underline{h}))\|f\|_\infty \|\underline{h}\|_\infty$,
(e) if $\|f\|_2 < \frac{p-1}{2}$, then $\|f\|_\infty \leq \|f\|_2 \leq \sqrt{\mu(f)}\|f\|_\infty$, and
(f) if $\|\underline{h}\|_2 < \frac{p-1}{2}$, then $\|\underline{h}\|_\infty \leq \|\underline{h}\|_2 \leq \sqrt{\ell\mu(\underline{h})}\|\underline{h}\|_\infty$.

The ring $\mathcal{R}$ is an integral domain, so if any $f, g \in \mathcal{R}$ satisfy $fg = 0$, then $f = 0$ or $g = 0$. On the other hand, $\mathcal{R}_\mathfrak{p}$ is not an integral domain. In fact, we have many elements of $f \in \mathcal{R}$ such that $f \equiv 0 (\text{mod } p)$ but such that $f \neq 0$ (e.g. one such element is $f(X) = pX$). However, $\mathcal{R}_\mathfrak{p}$ does satisfy the following property: elements of $\mathcal{R}_\mathfrak{p}$ with sufficiently small norm behave like elements of an integral domain with respect to each other. We formalize this notion with the following.

Lemma 1. *If $h \in \mathcal{R}$ has coset $\overline{h} \in \mathcal{R}_\mathfrak{p}$ such that $\left\|\overline{h}\right\|_\infty < \frac{p-1}{2}$ and $\overline{h} \equiv 0 (\text{mod } p)$, then $h = 0$.*

Proof. Write $h = \sum_{i \in [d]} h_i X^i$ with each $h_i \in \mathbb{Z}$. Since $\|h\|_\infty < \frac{p-1}{2}$, each $h_i \in \mathbb{Z} \cap [-\frac{p-1}{2}, \frac{p-1}{2}]$. Then $h (\text{mod } p) = \sum_{i \in [d]} \overline{h}_i X^i$ where each $\overline{h}_i$ is a coset with a representative equivalent to h_i modulo p and taken from the set $\mathbb{Z} \cap [-\frac{p-1}{2}, \frac{p-1}{2}]$. This representative is unique in this set, and h_i already represents this set, so the representative of each $\overline{h}_i$ is exactly h_i. Thus, each $h_i = 0$.

Corollary 1. *For any $f, g \in \mathcal{R}_\mathfrak{p}$ such that $\min(d, \mu(f), \mu(g))\|f\|_\infty \|g\|_\infty < \frac{p-1}{2}$, if $fg = 0$ then $f = 0$ or $g = 0$.*

Weighted and bounded subsets of $\mathcal{R}_\mathfrak{p}$ are close enough to integral domains to play a distinguished role in our analysis. We use the following notation (where $\mathbb{W}$ is for weighted-and-bounded subset).

$$\mathbb{W}(\mathcal{R}_\mathfrak{p}, \beta, \omega) = \{f \in \mathcal{R}_\mathfrak{p} \mid \|f\|_\infty \leq \beta, \mu(f) \leq \omega\}$$
$$\mathbb{W}(\mathcal{V}_\mathfrak{p}, \beta, \omega) = \left\{\underline{f} \in \mathcal{V}_\mathfrak{p} \mid \left\|\underline{f}\right\|_\infty \leq \beta, \mu(\underline{f}) \leq \omega\right\}$$

Partition these by Hamming weights into disjoint subsets $\mathbb{W}(\mathcal{R}_{\mathfrak{p}}, \beta, \omega) = \cup_{i=0}^{\omega} \widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \beta, i)$ and $\mathbb{W}(\mathcal{V}_{\mathfrak{p}}, \beta, \omega) = \cup_{i=0}^{\omega} \widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \beta, i)$ where we define $\widetilde{\mathbb{W}}(-, \beta, \omega)$ as follows.

$$\widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \beta, \omega) = \left\{ f \in \mathcal{R}_{\mathfrak{p}} \mid \|f\|_{\infty} \leq \beta, \mu(f) = \omega \right\}$$
$$\widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \beta, \omega) = \left\{ \underline{f} \in \mathcal{V}_{\mathfrak{p}} \mid \left\|\underline{f}\right\|_{\infty} \leq \beta, \mu(\underline{f}) = \omega \right\}$$

Then each $|\widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \beta, \omega)| = \binom{d}{\omega}(2\beta)^{\omega}$ and $|\widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \beta, \omega)| = |\widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \beta, \omega)|^{\ell}$. Also, since these unions are disjoint, we have $|\mathbb{W}(\mathcal{R}_{\mathfrak{p}}, \beta, \omega)| = \sum_{i=0} \binom{d}{i}(2\beta)^{i}$.

2.4 Hard Games and Hardness Estimation

The Ring Short Integer Solution problem with respect to a norm $\|\cdot\|$ over $\mathcal{R}$ is as follows [3].

Definition 5 ($(e, p, \ell, \|\cdot\|, \beta)$-RSIS game w.r.t. $\|\cdot\|$). *Let $\lambda \in \mathbb{N}$, let $e, p, \ell, \beta \in \mathsf{poly}(\lambda)$, and let $\mathcal{A}$ be any PPT algorithm. The following defines the $(e, p, \ell, \|\cdot\|, \beta)$-RSIS game with respect to $\|\cdot\|$.*

1. *The challenger and the player agree upon λ. The challenger computes $e(\lambda)$, $p(\lambda)$, $\ell(\lambda)$, and $\beta(\lambda)$, samples $\underline{a} \leftarrow \mathcal{V}_{\mathfrak{p}}$, and sets $\rho_{\mathsf{RSIS}} = (e, p, \ell, \beta, \underline{a})$. The challenger sends ρ_{RSIS} to $\mathcal{A}$.*
2. *The player $\mathcal{A}$ eventually outputs some $\underline{f} \in \mathcal{V}_{\mathfrak{p}}$.*

The player $\mathcal{A}$ succeeds if and only if $\underline{f} \neq 0$, $\left\|\underline{f}\right\| \leq \beta$, and $\langle \underline{a}, \underline{f} \rangle = 0$. If $\mathcal{A}$ runs in time at most t and succeeds at this game with probability at least ϵ, we say $\mathcal{A}$ is a (t, ϵ)-solver of the $(e, p, \ell, \|\cdot\|, \beta)$-RSIS game w.r.t. $\|\cdot\|$.

The $(e, p, \ell, \|\cdot\|, \beta)$-RSIS hardness assumption is that every (t, ϵ)-solver of the $(e, p, \ell, \|\cdot\|, \beta)$-RSIS game with some $t \in \mathsf{poly}(\lambda)$ also has $\epsilon \in \mathsf{negl}(\lambda)$, which we state more formally in the next section. This assumption is not valid for all parameters (e, p, ℓ, β), because if $\beta \geq \frac{p-1}{2}$, then Gaussian elimination can solve the problem in polynomial time.

Note that if $\left\|\underline{f}\right\|_{\infty} \leq \beta < \frac{p-1}{2}$ and $\mu(\underline{f}) \leq \omega$, then $\left\|\underline{f}\right\|_{2} \leq \sqrt{\ell\omega}\beta \leq \sqrt{\ell d}\beta$ thanks to Property (f). A solution to the $(e, p, \ell, \|\cdot\|_{\infty}, \beta)$-RSIS game with Hamming weight ω is therefore also a solution to the $(e, p, \ell, \|\cdot\|_{2}, \sqrt{\ell\omega}\beta)$-RSIS game (which is, in turn, a solution to the $(e, p, \ell, \|\cdot\|_{2}, \sqrt{\ell d}\beta)$-RSIS game). Hence, bounding the difficulty of the $(e, p, \ell, \|\cdot\|_{2}, \sqrt{\ell d}\beta)$-RSIS game is sufficient to also bound the difficulty of the $(e, p, \ell, \|\cdot\|_{\infty}, \beta)$-RSIS game.

To estimate hardness of the $(e, p, \ell, \|\cdot\|_{2}, \sqrt{d}\beta)$-RSIS game, we follow [4]. We estimate hardness by using bit-hardness, i.e. if $\mathcal{A}$ is a PPT $(t_{\mathcal{A}}, \epsilon_{\mathcal{A}})$-solver of a problem $\mathcal{P}$, we say the bits of hardness of $\mathcal{P}$ is $\inf_{\mathcal{A}} \log_2(\frac{t_{\mathcal{A}}}{\epsilon_{\mathcal{A}}})$, where this infimum is taken over all such PPT algorithms $\mathcal{A}$. We are aware that bit-hardness is not the best objective measure of a system's security, given different attack models and costs, but we use bit-hardness as a stand-in until more cryptanalysts can contribute. We assume the attacker uses a BKZ-like algorithm to look for a basis for the challenge lattice that has a sufficiently small root Hermite factor δ, solving the $(e, p, \ell, \|\cdot\|_{2}, \sqrt{\ell d}\beta)$-RSIS game. This δ is as follows except with negligible probability.

$$\delta = \left(\frac{\sqrt{\ell d}\beta}{p^{1/\ell}} \right)^{1/(\ell d - 1)}$$

We assume the attacker uses an algorithm similar to, but not identical to, the best known (quantum sieve method) implementation of the lattice reduction algorithm known as BKZ to find that basis.

The BKZ algorithm is parameterized by a block size $2^6 \leq b$ and is run by computing some k tours of the lattice. Each tour of the lattice takes time $2^{0.265b+16.4}$ operations in the conservative case where the attacker is the best known quantum SVP solver and enjoys Grover-like speed-ups (see [5], [4] for justifications for these numbers). After each tour, the BKZ algorithm can output a lattice basis with non-decreasing quality. The quality of this basis, say $\hat{\delta}$, asymptotically approaches the following as the number of tours increases.

$$\lim_{k\to\infty} \hat{\delta} \to \left(\frac{b\,(\pi b)^{\frac{1}{b}}}{2\pi e}\right)^{\frac{1}{2(b-1)}}$$

Usually, the output basis converges quickly in quality, so the BKZ algorithm is often terminated early, usually after some $k \geq 8\ell d$ tours, and the resulting lattice is often close to optimal in terms of its root Hermite factor.
We still assume that the attacker's algorithm is still parameterized by a block size $2^6 \leq b$. However, we conservatively assume that the attacker takes exactly one tour of the lattice, and does so in as few as $2^{0.265b}$ operations. We also conservatively assume that the resulting basis always has optimal quality exactly $\left(\frac{b(\pi b)^{\frac{1}{b}}}{2\pi e}\right)^{\frac{1}{2(b-1)}}$. This way, if $\beta < \frac{p-1}{2}$ and $\left(\frac{\sqrt{\ell d}\beta}{p^{1/\ell}}\right)^{1/(\ell d-1)} < \left(\frac{b(\pi b)^{\frac{1}{b}}}{2\pi e}\right)^{\frac{1}{2(b-1)}}$ then the attacker fails to output a lattice basis of sufficient quality to solve the $(e, p, \ell, \|\cdot\|_2, \sqrt{\ell d}\beta)$-RSIS game (and therefore the $(e, p, \ell, \|\cdot\|_\infty, \beta)$-RSIS game).

Assumption 1. *Let $\lambda \in \mathbb{N}$. If $\beta < \frac{p-1}{2}$ and*

$$\left(\frac{\sqrt{\ell d}\beta}{p^{1/\ell}}\right)^{1/(\ell d-1)} < \left(\frac{\lambda\left(\pi \cdot \frac{\lambda}{0.265}\right)^{\frac{0.265}{\lambda}}}{2\pi e(0.265)}\right)^{\frac{1}{2(\frac{\lambda}{0.265}-1)}} \tag{2.1}$$

then solving the $(e, p, \ell, \|\cdot\|_\infty, \beta)$-RSIS game has at least λ bits of hardness.

2.5 The Forking Lemma

As in [1] we use the following variant of the Forking Lemma.

Lemma 2 (Rewinding Lemma). *Let S, R, T be finite, non-empty sets, consider any function $f : S\times R\times T \to \{0,1\}$. Let X, Y, Y', Z, Z' be mutually independent random variables where $\mathrm{Supp}(X) = S$, Y and Y' are uniformly distributed over R, and $\mathrm{Supp}(Z) = \mathrm{Supp}(Z') = T$. Let $\epsilon = \mathbb{P}[f(X,Y,Z) = 1]$. Then $\mathbb{P}[f(X,Y,Z) = 1 \wedge f(X,Y',Z') = 1 \wedge Y \neq Y'] \geq \epsilon^2 - \frac{\epsilon}{|R|}$.*

As in [1], we define simulation algorithms as follows.

Definition 6 (Simulation Algorithm). *Let $Q \in \mathbb{N}$, and let $\mathcal{X}, \mathcal{H}$ be sets such that $|\mathcal{H}| > 1$. We say a randomized algorithm $\mathcal{S}$ is a simulation algorithm if $\mathcal{S}$ inputs $x \in \mathcal{X}$ and $\underline{h} \in \mathcal{H}^Q$ and outputs $\mathsf{out} = (i, \mathsf{aux}) \in (\{\bot\}\cup[Q])\times\{0,1\}^*$. For any distribution F over $\mathcal{X}$, we define the advantage of $\mathcal{S}$ as $\mathbb{P}\left[(i, \mathsf{aux}) \leftarrow \mathcal{S}(x, \underline{h}) \wedge i \neq \bot \mid x \leftarrow F, \underline{h} \leftarrow \mathcal{H}^Q\right]$. We denote the advantage with $\mathsf{Adv}_{\mathcal{S}}$.*

Given a simulation algorithm $\mathcal{S}$, we fork $\mathcal{S}$ by running the following algorithm.

Definition 7 (Forking Algorithm). *Let $Q \in \mathbb{N}$, let $\mathcal{X}, \mathcal{H}$ be sets such that $|\mathcal{H}| > 1$, and let $\mathcal{S}$ be a simulation algorithm as defined above. The forking algorithm for $\mathcal{S}$, denoted $\mathsf{Fork}_{\mathcal{S}}$, inputs $x \in \mathcal{X}$, and works as follows.*

1. *Sample random coins τ for $\mathcal{S}$.*
2. *Sample $\underline{h} = (\underline{h}_0, \underline{h}_1, \underline{h}_2, \underline{h}_3, \ldots, \underline{h}_{Q-1}) \leftarrow\!\!\$\, \mathcal{H}^Q$.*
3. *Compute $(i, \mathsf{aux}) \leftarrow \mathcal{S}(x, \underline{h})$.*
4. *If $i = \perp$, output $(\perp, \perp, \perp)$ and terminate.*
5. *Otherwise, sample $\underline{h}' = (\underline{h}'_0, \underline{h}'_1, \underline{h}'_2, \underline{h}'_3, \ldots, \underline{h}'_{Q-1}) \leftarrow\!\!\$\, \mathcal{H}^Q$. For each $j \in [i]$, set $\underline{h}^*_j = \underline{h}_j$ if $j < i$, and for each $j \in [Q] \setminus [i]$, set $\underline{h}^*_j = \underline{h}'_j$.*
6. *Compute $(i', \mathsf{aux}') \leftarrow \mathcal{S}(x, \underline{h}^*)$.*
7. *If $i = i'$ and $\underline{h}_i \neq \underline{h}'_i$, output $(1, \mathsf{aux}, \mathsf{aux}')$ and terminate.*
8. *Otherwise, output $(\perp, \perp, \perp)$ and terminate.*

Moreover, for a distribution F over $\mathcal{X}$, we say the advantage of $\mathsf{Fork}_{\mathcal{S}}$ is the probability that the output leads with a 1, i.e. $\mathbb{P}\left[(b, \mathsf{aux}, \mathsf{aux}') \leftarrow \mathsf{Fork}_{\mathcal{S}}(x) \wedge b = 1 \mid x \leftarrow\!\!\$\, F\right]$.

The rewinding lemma implies the following.

Lemma 3 (General Forking Lemma). *Let $Q \in \mathbb{N}$, and let $\mathcal{X}, \mathcal{H}$ be sets such that $|\mathcal{H}| > 1$, let F be a distribution over $\mathcal{X}$, and let $\mathcal{S}$ be a simulation algorithm with advantage $\mathsf{Adv}_{\mathcal{S}}$. Then $\mathsf{Adv}_{\mathsf{Fork}_{\mathcal{S}}} \geq \mathsf{Adv}_{\mathcal{S}} \left(\frac{\mathsf{Adv}_{\mathcal{S}}}{Q} - \frac{1}{|\mathcal{H}|} \right)$.*

Later, we assume an attacker is as powerful as possible and therefore only needs one query ($Q = 1$) to do its job.

2.6 Tightness Gap

The forking lemma provides a tightness gap as described in [4] and [6]. If there exists a reduction from problem $\mathcal{Q}$ to problem $\mathcal{P}$, given an algorithm $\mathcal{A}$ which is a $(t_{\mathcal{A}}, \epsilon_{\mathcal{A}})$-solver for $\mathcal{P}$, we can build an algorithm $\mathcal{B}$ which is a $(t_{\mathcal{B}}, \epsilon_{\mathcal{B}})$-solver for $\mathcal{Q}$ which works by running $\mathcal{A}$ as a subroutine. If $\mathcal{Q}$ is thought to be hard, then $\mathcal{P}$ will likewise be hard. Since $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, $t_{\mathcal{B}} \geq t_{\mathcal{A}}$. Since the success of $\mathcal{B}$ requires the success of $\mathcal{A}$, $\epsilon_{\mathcal{B}} \leq \epsilon_{\mathcal{A}}$.

Definition 8. *A reduction taking algorithm $\mathcal{A}$ to algorithm $\mathcal{B}$ has tightness gap $\gamma = \frac{t_{\mathcal{B}} \epsilon_{\mathcal{A}}}{t_{\mathcal{A}} \epsilon_{\mathcal{B}}}$.*

To ensure that the problem $\mathcal{P}$ has λ bits of security, we target $\mathcal{Q}$ to have $\lambda + \log_2(\gamma)$ bits of security.

3 Fusion Signatures

In subsection 3.1, we define a modified version of the signature scheme from [1]. In subsection 3.2, we present the associated security theorem statements, proofs for which can be found in the appendix.

3.1 Construction

We call the following scheme Π_F in the sequel.

Definition 9. *The Fusion aggregatable signature scheme is a tuple of the following algorithms.*

1. $\mathsf{Pgen}(\lambda) \to \rho$ *inputs $\lambda \in \mathbb{N}$ and outputs public parameters ρ containing the following data.*

- *(i)* *an exponent* $e \in \mathbb{N}$*, a prime* p*, and a module rank* $\ell \in \mathbb{N}$*,*
- *(ii)* *an aggregation capacity* $K \in \mathbb{N}$ *with* $K \geq 2$*,*
- *(iii)* *bounds* $\beta_{sk}, \beta_{ch}, \beta_{ag}, \beta_v \in \mathbb{N}$*,*
- *(iv)* *weights* $\omega_{sk}, \omega_{ch}, \omega_{ag}, \omega_v \in \mathbb{N}$*,*
- *(v)* *collision-resistant hash function* $\mathsf{H}_{ch} : \{0,1\}^* \to \widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ch}, \omega_{ch})$*,*
- *(vi)* *collision-resistant hash function* $\mathsf{H}_{ag} : \{0,1\}^* \to \widetilde{\mathbb{W}}^K(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ag}, \omega_{ag})$*, and*
- *(vii)* *a new random module element* $\underline{a} \leftarrow\!\!\$\ \mathcal{V}_{\mathfrak{p}}$*.*

2. $\mathsf{KGen}(\rho) \to (\mathsf{sk}, \mathsf{vk})$ *inputs parameters* ρ *and outputs a keypair* $(\mathsf{sk}, \mathsf{vk})$ *computed as follows.*
   - *(a)* *Sample* $\underline{f}_0, \underline{f}_1 \leftarrow\!\!\$\ \widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \infty, \beta_{sk}, \omega_{sk})$*.*
   - *(b)* *Compute* $g_i = \langle \underline{a}, \underline{f}_i \rangle$ *for* $i = 0, 1$*.*
   - *(c)* *Set the secret signing key* $\mathsf{sk} = (\underline{f}_0, \underline{f}_1)$ *and the public verification key* $\mathsf{vk} = (g_0, g_1)$*.*
   - *(d)* *Output* $(\mathsf{sk}, \mathsf{vk})$*.*
3. $\mathsf{Sign}(\rho, \mathsf{sk}, m) \to \xi$ *inputs parameters* ρ*, a keypair* $(\mathsf{sk}, \mathsf{vk})$*, and a message* m*, and outputs a signature* ξ *as follows.*
   - *(a)* *Compute the signature challenge* $c = \mathsf{H}_{ch}(\mathsf{vk}, m)$*.*
   - *(b)* *Set* $\xi = \underline{f}_0 c + \underline{f}_1$*.*
   - *(c)* *Output* ξ*.*
4. $\mathsf{Agg}(\rho, (\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]}) \to \xi_{ag}$ *inputs parameters* ρ *and a list of public verification key-message-signature triples* $(\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]}$ *and outputs an aggregate signature* ξ_{ag} *as follows.*
   - *(a)* *Sort the list of triples* $(\mathsf{vk}_i, m_i, \xi_i)_{i \in [N]}$ *by the public verification keys* vk_i*.*
   - *(b)* *For each* $i \in [N]$*, compute the signature challenge* $c_i = \mathsf{H}_{ch}(\mathsf{vk}_i, m_i)$*.*
   - *(c)* *Compute the aggregation coefficients* $(\alpha_i)_{i \in [N]} = \underline{\alpha} \leftarrow \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i \in [N]})$*.*
   - *(d)* *Compute* $\xi_{ag} = \sum_{i \in [N]} \alpha_i \xi_i$*.*
   - *(e)* *Output* ξ_{ag}*.*
5. $\mathsf{AggVf}(\rho, (\mathsf{vk}_i, m_i)_{i \in [N]}, \xi_{ag}) \to \{0,1\}$ *inputs parameters* ρ*, a list of public verification key-message pairs* $(\mathsf{vk}_i, m_i)_{i \in [N]}$*, and an aggregate signature* ξ_{ag}*, and outputs a bit* $b \in \{0,1\}$ *as follows.*
   - *(a)* *If any* $\mathsf{vk}_i \notin \mathcal{R}_{\mathfrak{p}}^2$ *or any* vk_i *appears more than once or* $\|\xi_{ag}\|_\infty > \beta_v$ *or* $\mu(\xi_{ag}) > \omega_v$*, output* 0 *and terminate.*
   - *(b)* *Otherwise, sort the input list of pairs* $(\mathsf{vk}_i, m_i)_{i \in [N]}$ *by the public verification keys* vk_i*.*
   - *(c)* *For each* $i \in [N]$*, compute the signature challenge* $c_i = \mathsf{H}_{ch}(\mathsf{vk}_i, m_i)$*.*
   - *(d)* *Compute the aggregation coefficients* $(\alpha_i)_{i \in [N]} = \underline{\alpha} \leftarrow \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i \in [N]})$*.*
   - *(e)* *Compute* $\mathsf{targ} = \sum_{i \in [N]} \alpha_i (g_{i,0} c_i + g_{i,1})$ *where each* $\mathsf{vk}_i = (g_{i,0}, g_{i,1})$*.*
   - *(f)* *If* $\langle \underline{a}, \xi_{ag} \rangle = \mathsf{targ}$*, output* 1 *and terminate.*
   - *(g)* *Otherwise, output* 0 *and terminate.*

3.2 Properties of Fusion Signatures

The output ξ from Sign has $\|\xi\|_\infty \leq \beta_{sk}(1 + \min(d, \omega_{sk}, \omega_{ch})\beta_{ch})$ and $\mu(\xi) \leq \min(d, \omega_{sk}(1 + \omega_{ch}))$. For clarity in the sequel, define the following.

$$\omega_v' = \min(d, \omega_{sk}(1 + \omega_{ch})) \tag{3.1}$$

$$\beta_v' = \beta_{sk}(1 + \min(d, \omega_{sk}, \omega_{ch})\beta_{ch}) \tag{3.2}$$

Note that both Agg and AggVf sort their inputs. This is important to ensure that aggregation and verification are not sensitive to key order. As in many one-time signature schemes, revealing two or more signatures for any keypair $(\mathsf{sk}, \mathsf{vk})$ reveals enough information to compute sk, and should be avoided at all costs. We use the following security theorems, proven in the appendices.

Theorem 2. *If* $\min(d, K\omega_{ag}\omega'_v) \leq \omega_v$ *and* $K\min(d, \omega_{ag}, \omega'_v)\beta_{ag}\beta'_v \leq \beta_v$*, then* $\mathit{\Pi}_F$ *is correct.*

Theorem 3. *If* $\mathit{\Pi}_F$ *is correct, then it is compact.*

Theorem 4. *Let* $\underline{a} \in \mathcal{V}$*,* $\beta'_v < \frac{p-1}{2}$*, and* $c \in \widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ch}, \omega_{ch})$*. If* $(\underline{f}_0, \underline{f}_1) \leftarrow\!\!\$\ \widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \infty, \beta_{sk}, \omega_{sk})^2$ *and* $2^\lambda p^{2d}(2\beta'_v + 1)^{\ell d} \leq \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$ *, then*

$$\mathbb{P}\left[\exists(\underline{f}'_0, \underline{f}'_1) \neq (\underline{f}_0, \underline{f}_1) : \langle \underline{a}, \underline{f}_0 \rangle = \langle \underline{a}, \underline{f}'_0 \rangle, \langle \underline{a}, \underline{f}_1 \rangle = \langle \underline{a}, \underline{f}'_1 \rangle, \underline{f}_0 c + \underline{f}_1 = \underline{f}'_0 c + \underline{f}'_1\right] \geq 1 - 2^{-\lambda}.$$

Theorem 5. *Let* $\underline{a} \in \mathcal{V}_{\mathfrak{p}}$*,* $(c, c') \in \widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ch}, \omega_{ch})^2$ *such that* $c \neq c'$*,* $(\xi, \xi') \in \mathbb{W}(\mathcal{V}_{\mathfrak{p}}, \infty, \beta'_v, \omega'_v)^2$*, and* $\alpha \in \widetilde{\mathbb{W}}(\mathcal{R}_{\mathfrak{p}}, \infty, 2\beta_{ag}, 2\omega_{ag})$*. If* $8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} < \frac{p-1}{2}$ *then there is at most a single pair* $(\underline{f}_0, \underline{f}_1) \in \mathbb{W}(\mathcal{V}_{\mathfrak{p}}, \infty, \beta_{sk}, \omega_{sk})^2$ *such that* $\xi = \underline{f}_0 c + \underline{f}_1$ *and* $\xi' = \alpha(\underline{f}_0 c' + \underline{f}_1)$*.*

Theorem 6. *Let* ϵ_{ag}*,* $\epsilon_{ch}, \epsilon_{\mathcal{A}}, \epsilon' \in (0, 1)$*,* $t, t_0, t_1, t_2, t' \geq 0$*, and* $\omega, \beta \in \mathbb{N}$ *such that all the following hold.*

$$\begin{aligned}
\epsilon_{ag} =& \begin{pmatrix} d \\ \omega_{ag} \end{pmatrix}^{-1} (2\beta_{ag})^{-\omega_{ag}} \\
\epsilon_{ch} =& \begin{pmatrix} d \\ \omega_{ch} \end{pmatrix}^{-1} (2\beta_{ch})^{-\omega_{ch}} \\
t' =& 2(t + t_0) + t_1 + t_2 \\
\epsilon' =& \frac{1}{2}\left(1 - \epsilon_{ch}\right)^2 \left(1 - \epsilon_{ag}\right)^2 \left(1 - \frac{\epsilon_{ag}}{(1 - \epsilon_{ch})(1 - \epsilon_{ag})\epsilon_{\mathcal{A}}}\right) \epsilon_{\mathcal{A}}^2 \\
\omega \geq& \min(d, 2\omega_v + 2\omega_{ag}\omega'_v) \\
\beta \geq& 2\beta_v + 2\min(d, 2\omega_{ag}, \omega'_v)\beta_{ag}\beta'_v
\end{aligned}$$

If (i) $\mathit{\Pi}_F$ *is correct and (ii) the hypotheses of Theorems 4 and 5 hold and (iii) there exists some PPT algorithm* $\mathcal{A}$ *is a* $(t, \epsilon_{\mathcal{A}})$*-forger of Definition 4 for* $\mathit{\Pi}_F$ *which makes at most* 1 *query to* H_{ag} *and at most* K *queries to* H_{ch}*, then there exists an algorithm* $\mathcal{B}$ *that is a* (t', ϵ')*-solver of the* $(e, p, \ell, \|\cdot\|_\infty, \beta)$*-RSIS game of Definition 5 whose solution to that game has Hamming weight at most* ω*.*

Next consider the tightness gap in Lemma 4.

Lemma 4. *Let* $\lambda \in \mathbb{N}$*,* $\kappa \in \mathbb{N}_0$ *and assume forging a signature from* $\mathit{\Pi}_F$ *has* λ *bits of hardness. If all of the following conditions hold then the proof of Theorem 6 implies a tightness gap* $\log_2(\gamma) \leq 9 + \lambda$*.*

- $t_0 < t$,
- $t_1 + t_2 < 2(t + t_0)$,
- $\epsilon_{ch} < 2^{-\lambda}$,
- $\epsilon_{ag} < \frac{1}{2}$,
- $\frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \leq 2^{-(\lambda+1)}$.

Note that the first two assumptions in Lemma 4 are reasonable, because t_0, t_1, and t_2 are all simple computations, even for classical computers. The remaining assumptions boil down to assuming that, if we have cryptographic hardness against forgery, then we have a cryptographic number of aggregation coefficients and challenges.

4 Parameter Selection

In Table 4, we provide some recommended parameter sets. To see how we obtained these parameter sets, we consider a list of the constraints required to satisfy Assumption 1, Theorem 2, Theorem 3, Theorem 4, Theorem 5, Theorem 6, and Lemma 4. Recall our definition of ω_v' and β_v' in Equations (3.1) and (3.2), and set the following.

$$\omega_v = \min(d, K\omega_{ag}\omega_v') \tag{4.1}$$
$$\beta_v = K\min(d, \omega_{ag}, \omega_v')\beta_{ag}\beta_v' \tag{4.2}$$
$$\omega = \min(d, 2\omega_v + 2\omega_{ag}\omega_v') \tag{4.3}$$
$$\beta = 2\beta_v + 2\min(d, 2\omega_{ag}, \omega_v')\beta_{ag}\beta_v' \tag{4.4}$$

We force the ring $\mathcal{R}_{\mathfrak{p}}$ to be friendly to the number-theoretic transform for faster signing, aggregation, and verification. For our unforgeability proof to go through, we need to satisfy the hypotheses of Lemma 4 by picking parameters allowing for the existence of some ℓ such that $2^\lambda p^{2d}(2\beta_v'+1)^{\ell d} < \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$. Note that $2^\lambda p^{2d} > 1$. Hence, if $\frac{\binom{d}{\omega_{sk}}^2(2\beta_{sk})^{2\omega_{sk}}}{(2\beta_v'+1)^d} \leq 1$, then no ℓ will allow us to satisfy the hypotheses of Lemma 4. On the other hand, if $\frac{\binom{d}{\omega_{sk}}^2(2\beta_{sk})^{2\omega_{sk}}}{(2\beta_v'+1)^d} > 1$, then there always exists an ℓ that allows us to satisfy the hypotheses of Lemma 4. Bringing all our parameter requirements together, we seek parameters such that all the following hold.

1. d is a power of two, p is a prime, and $p - 1 \equiv 0 (\mathrm{mod}\ 2d)$ (NTT-friendliness),
2. $\beta < \frac{p-1}{2}$ (from Assumption 1),
3. $8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} < \frac{p-1}{2}$ (from Lemma 5),
4. $\left(\frac{\sqrt{\ell d}\beta}{p^{1/\ell}}\right)^{1/(\ell d-1)} < \left(\frac{(2\lambda+9)\left(\pi\cdot\frac{2\lambda+9}{0.265}\right)^{\frac{0.265}{2\lambda+9}}}{2\pi e(0.265)}\right)^{\left(2\left(\frac{2\lambda+9}{0.265}-1\right)\right)^{-1}}$ (from Assumption 1 and Theorem 4),
5. $(2\beta_v'+1)^d < \binom{d}{\omega_{sk}}^2(2\beta_{sk})^{2\omega_{sk}}$ (see above),
6. $2^\lambda p^{2d}(2\beta_v'+1)^{\ell d} \leq \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$ (from Lemma 4),
7. $\epsilon_{ch} = \binom{d}{\omega_{ch}}^{-1}(2\beta_{ch})^{-\omega_{ch}} < 2^{-\lambda}$ (from Theorem 4),
8. $\epsilon_{ag} = \binom{d}{\omega_{ag}}^{-1}(2\beta_{ag})^{-\omega_{ag}} < \frac{1}{2}$ (from Theorem 4),
9. $\frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} < 2^{-(\lambda+1)}$ (from Theorem 4).

We describe in section B the heuristic search method we employed to find parameters good efficiency with respect to per-signer space complexity. With a tightness gap is about $\log(\gamma) \approx \lambda+9$, we want the underlying RSIS game to have about $\lambda + \log(\gamma) \approx 2\lambda+9$ bits of difficulty. In subsection C.1, subsection C.2, and subsection C.3, we show they satisfy all the requirements (1) through (9).

A Security Theorems and Proofs

Theorem 2. *If* $\min(d, K\omega_{ag}\omega_v') \leq \omega_v$ *and* $K\min(d, \omega_{ag}, \omega_v')\beta_{ag}\beta_v' \leq \beta_v$*, then* Π_F *is correct.*

Proof. In the event $E(\Pi_F, \lambda)$ we can parse each $\mathsf{vk}_i = (g_{i,0}, g_{i,1})$, parse each $\mathsf{sk}_i = (\underline{f}_{i,0}, \underline{f}_{i,1})$, and compute $c_i = \mathsf{H}_{ch}(\mathsf{vk}_i, m_i)$ for each $i \in [N]$. Using these values, each $\xi_i = \underline{f}_{i,0}c_i + \underline{f}_{i,1}$ by specification of Sign. Moreover, the list $(\mathsf{vk}_i, m_i, c_i)_{i\in[N]}$ is sorted by the keys vk_i in $E(\Pi_F, \lambda)$. Let $\underline{\alpha} = (\alpha_{i\in[N]}) = \mathsf{H}_{ag}((\mathsf{vk}_i, m_i, c_i)_{i\in[N]})$ and $\xi_{ag} = \sum_{i\in[N]}\alpha_i(\underline{f}_{i,0}c_i + \underline{f}_{i,1})$ by specification

	Fusion Light	Fusion Mid		Fusion Heavy	
Parameter	Value				
λ	128	128	256	128	256
p	2147465729				
d	64	128		256	
K	1796	20813	236	32417	2818
ℓ	195	97	166	48	83
ω_{ch}	27	31	53	23	60
β_{ch}	3	1	3	1	1
ω_{ag}	35	31	67	23	60
β_{ag}	2	1	2	1	1
ω_{sk}	64	128		256	
β_{sk}	52	26	105	30	52
β_v	536070080	536808896	531283200	536825520	536321760
Statistic	Value				
$\lvert vk\rvert$ (KB)	0.496	0.992	0.992	1.984	1.984
$\lvert\xi\rvert$ (KB)	21.84	17.072	42.496	16.896	34.528
$\lvert\xi\rvert_{ag}$ (KB)	46.8	46.56	79.68	46.08	79.68
$\lvert\mathsf{vk}\rvert + \lvert\xi_{ag}\rvert/K$ (KB)	0.523	0.995	1.330	1.986	2.013
N to beat CRYSTALS	25	33	56	106	183
N to beat Falcon	276	No	No	No	No

Table 1. Parameters generated by the heuristic presented in section B. Scheme names are selected based on per-signer space complexity, not security.

of Agg. Moreover, $g_{i,0} = \langle \underline{a}, \underline{f}_{i,0} \rangle$ and $g_{i,1} = \langle \underline{a}, \underline{f}_{i,1} \rangle$ for each $i \in [N]$ by specification of KGen. Inner products are linear, so $\langle \underline{a}, \xi_{ag} \rangle = \sum_{i \in [N]} \alpha_i (g_{i,0} c_i + g_{i,1})$ exactly.

Hence, it is sufficient to check that ξ_{ag} has $\|\xi_{ag}\|_\infty \leq \beta_v$ and $\mu(\xi_{ag}) \leq \omega_v$. First, consider the Hamming weight. To compute $\mu(\xi_{ag})$, consider $\mu(\xi_i)$ for an honestly computed ξ_i.

$$\begin{aligned}
\mu(\underline{f}_{i,0} c_i + \underline{f}_{i,1}) \leq & \min(d, \mu(\underline{f}_{i,0} c_i) + \mu(\underline{f}_{i,1})) \\
\leq & \min(d, \min(d, \mu(\underline{f}_{i,0}) \mu(c_i)) + \omega_{sk}) \\
\leq & \min(d, \min(d, \omega_{sk} \omega_{ch}) + \omega_{sk}) \\
= & \min(d, \omega_{sk}(1 + \omega_{ch})) \\
\leq & \omega_v' \\
\mu(\xi_{ag}) = & \mu \left(\sum_{i \in [N]} \alpha_i (\underline{f}_{i,0} c_i + \underline{f}_{i,1}) \right) \\
\leq & \min \left(d, \sum_{i \in [N]} \mu \left(\alpha_i (\underline{f}_{i,0} c_i + \underline{f}_{i,1}) \right) \right) \\
\leq & \min \left(d, \sum_{i \in [N]} \min \left(d, \mu(\alpha_i) \mu \left(\underline{f}_{i,0} c_i + \underline{f}_{i,1} \right) \right) \right) \\
\leq & \min \left(d, \sum_{i \in [N]} \min \left(d, \omega_{ag} \omega_v' \right) \right) \\
\leq & \min \left(d, N \min \left(d, \omega_{ag} \omega_v' \right) \right) \\
= & \min(d, N \omega_{ag} \omega_v') \\
\leq & \min(d, K \omega_{ag} \omega_v') \\
\leq & \omega_v
\end{aligned}$$

Next consider the norm. Since $\beta_v' = \min(d, \omega_{sk}, \omega_{ch}) \beta_{sk} (1 + \beta_{ch})$ and $K \min(d, \omega_{ag}, \omega_v') \beta_{ag} \beta_v' \leq \beta_v$, we have the following.

$$\begin{aligned}
\|\xi_i\|_\infty = & \left\| \underline{f}_{i,0} c_i + \underline{f}_{i,1} \right\|_\infty \\
\leq & \left\| \underline{f}_{i,0} c_i \right\|_\infty + \left\| \underline{f}_{i,1} \right\|_\infty \\
\leq & \min(d, \mu(\underline{f}_{i,0}), \mu(c_i)) \left\| \underline{f}_{i,0} \right\|_\infty \|c_i\|_\infty + \beta_{sk} \\
\leq & \min(d, \omega_{sk}, \omega_{ch}) \beta_{sk} \beta_{ch} + \beta_{sk} \\
= & \beta_{sk} (1 + \min(d, \omega_{sk}, \omega_{ch}) \beta_{ch}) \\
= & \beta_v'
\end{aligned}$$

$$\begin{aligned}
\|\xi_{ag}\|_\infty =& \left\|\sum_{i\in[N]} \alpha_i(\underline{f}_{i,0}c_i + \underline{f}_{i,1})\right\|_\infty \\
\leq& \sum_{i\in[N]} \left\|\alpha_i(\underline{f}_{i,0}c_i + \underline{f}_{i,1})\right\|_\infty \\
\leq& \sum_{i\in[N]} \min\left(d, \mu(\alpha_i), \mu(\underline{f}_{i,0}c_i + \underline{f}_{i,1})\right) \|\alpha_i\|_\infty \left\|\underline{f}_{i,0}c_i + \underline{f}_{i,1}\right\|_\infty \\
\leq& \sum_{i\in[N]} \min(d, \omega_{ag}, \omega_v')\beta_{ag}\beta_v' \\
=& N\min(d, \omega_{ag}, \omega_v')\beta_{ag}\beta_v' \\
=& K\min(d, \omega_{ag}, \omega_v')\beta_{ag}\beta_v' \\
\leq& \beta_v
\end{aligned}$$

Theorem 3. *If Π_F is correct, then it is compact.*

Proof. It is sufficient to show that the information-theoretic minimum bits required to describe any $\xi_{ag} \in \Xi_{ag}$ is logarithmic in K. Since $\xi_{ag} \in \mathcal{V}_\mathfrak{p}$, ξ_{ag} is an ℓ-vector of polynomials, each with degree-bound d, where ℓ and d are not functions of K. Hence, if it takes M bits to describe a coefficient of one of these polynomials, then it takes ℓdM bits to describe ξ_{ag}. So it is sufficient to show that M is logarithmic in K. Of course, $\|\xi_{ag}\|_\infty$ is the absolute maximum of any such coefficient, so each coefficient is drawn from a set with $2\|\xi_{ag}\|_\infty + 1$ elements. As in Theorem 2, we have that $\|\xi_{ag}\|_\infty \leq K\min(d, \omega_{ag})\beta_{ag}\beta_v'$. All other variables are polynomial in λ and are not functions of K. This norm is linear in K and M is logarithmic in the norm, so M is logarithmic in K.

To prove unforgeability, we make use of Corollary 1 and the following lemmata.

Theorem 7. *Let $r \in (0,1)$ and let $\phi : X \to Y$ be any function between finite sets such that $|Y|/|X| \leq r$. Then $\mathbb{P}\left[\exists x' \in X : x' \neq x \wedge \phi(x') = \phi(x) \mid x \leftarrow\!\!\$\, X\right] \geq 1 - r$.*

Theorem 4. *Let $\underline{a} \in \mathcal{V}$, $\beta_v' < \frac{p-1}{2}$, and $c \in \widetilde{\mathbb{W}}(\mathcal{R}_\mathfrak{p}, \infty, \beta_{ch}, \omega_{ch})$. If $(\underline{f}_0, \underline{f}_1) \leftarrow\!\!\$\, \widetilde{\mathbb{W}}(\mathcal{V}_\mathfrak{p}, \infty, \beta_{sk}, \omega_{sk})^2$ and $2^\lambda p^{2d}(2\beta_v' + 1)^{\ell d} \leq \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$, then*

$$\mathbb{P}\left[\exists(\underline{f}_0', \underline{f}_1') \neq (\underline{f}_0, \underline{f}_1) : \langle\underline{a}, \underline{f}_0\rangle = \langle\underline{a}, \underline{f}_0'\rangle, \langle\underline{a}, \underline{f}_1\rangle = \langle\underline{a}, \underline{f}_1'\rangle, \underline{f}_0 c + \underline{f}_1 = \underline{f}_0' c + \underline{f}_1'\right] \geq 1 - 2^{-\lambda}.$$

Proof. We apply Theorem 7. For each $c \in \widetilde{\mathbb{W}}(\mathcal{R}_\mathfrak{p}, \infty, \beta_{ch}, \omega_{ch})$, define $\theta_c : \widetilde{\mathbb{W}}(\mathcal{V}_\mathfrak{p}, \infty, \beta_{sk}, \omega_{sk})^2 \to \mathcal{R}_\mathfrak{p}^2 \times \mathbb{W}(\mathcal{V}_\mathfrak{p}, \infty, \beta_v', d)$ by mapping $(\underline{f}_0, \underline{f}_1) \mapsto (\langle\underline{a}, \underline{f}_0\rangle, \langle\underline{a}, \underline{f}_1\rangle, \underline{f}_0 c + \underline{f}_1)$. Certainly by definition $|\mathrm{dom}(\theta_c)| = \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$ and $|\mathrm{cod}(\theta_c)| = p^{2d}(2\beta_v'+1)^{\ell d}$. The result follows directly from Theorem 7.

We can tighten the constraint from Lemma 4 to $2^\lambda p^{2d} \sum_{i=0}^{\omega_v'} \binom{d}{i}^\ell (2\beta_v')^{\ell i} < \binom{d}{\omega_{sk}}^{2\ell}(2\beta_{sk})^{2\ell\omega_{sk}}$ in the special case that $\omega_v' < d$. However, in practice, we only found parameters with $\omega_{sk} = \omega_v' = \omega_v = \omega = d$. It is possible that tightening the bounds like this can lead to better parameters than the ones we present here.

Theorem 5. *Let $\underline{a} \in \mathcal{V}_\mathfrak{p}$, $(c, c') \in \widetilde{\mathbb{W}}(\mathcal{R}_\mathfrak{p}, \infty, \beta_{ch}, \omega_{ch})^2$ such that $c \neq c'$, $(\xi, \xi') \in \mathbb{W}(\mathcal{V}_\mathfrak{p}, \infty, \beta_v', \omega_v')^2$, and $\alpha \in \widetilde{\mathbb{W}}(\mathcal{R}_\mathfrak{p}, \infty, 2\beta_{ag}, 2\omega_{ag})$. If $8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} < \frac{p-1}{2}$ then there is at most a single pair $(\underline{f}_0, \underline{f}_1) \in \mathbb{W}(\mathcal{V}_\mathfrak{p}, \infty, \beta_{sk}, \omega_{sk})^2$ such that $\xi = \underline{f}_0 c + \underline{f}_1$ and $\xi' = \alpha(\underline{f}_0 c' + \underline{f}_1)$.*

Proof. If there exists pairs $(\underline{f}_0, \underline{f}_1), (\underline{f}'_0, \underline{f}'_1) \in \mathbb{W}(\mathcal{V}_{\mathfrak{p}}, \infty, \beta_{sk}, \omega_{sk})^2$ such that $\xi = \underline{f}_0 c + \underline{f}_1 = \underline{f}'_0 c + \underline{f}'_1$ and $\xi' = \alpha(\underline{f}_0 c' + \underline{f}_1) = \alpha(\underline{f}'_0 c' + \underline{f}'_1)$, then we have the following.

$$\begin{aligned}(\underline{f}_0 - \underline{f}'_0)c + (\underline{f}_1 - \underline{f}'_1) =&0\\ \alpha((\underline{f}_0 - \underline{f}'_0)c' + (\underline{f}_1 - \underline{f}'_1)) =&0\end{aligned}$$

Therefore, $\alpha(c - c')(\underline{f}_0 - \underline{f}'_0) = 0$. Moreover, we have that $\xi - \underline{f}_0 c = \underline{f}_1$ and $\xi - \underline{f}'_0 c = \underline{f}'_1$. So, it is sufficient to show that $\underline{f}_0 = \underline{f}'_0$ in order to conclude that $\underline{f}_1 = \underline{f}'_1$. We apply Corollary 1 twice. If $\alpha(c - c')(\underline{f}_0 - \underline{f}'_0) = 0$, $\alpha \neq 0$, $c \neq c'$, and both of the following hold, then $\underline{f}_0 = \underline{f}'_0$.

1. $\min(d, \mu(\alpha), \mu((c - c')(\underline{f}_0 - \underline{f}'_0)))\|\alpha\|_\infty \left\|(c - c')(\underline{f}_0 - \underline{f}'_0)\right\|_\infty < \frac{p-1}{2}$, and
2. $\min(d, \mu(c - c'), \mu(\underline{f}_0 - \underline{f}'_0))\|c - c'\|_\infty \left\|\underline{f}_0 - \underline{f}'_0\right\|_\infty < \frac{p-1}{2}$

Consider this first condition. We have that $\mu(\alpha) \leq 2\omega_{ag}$ and $\|\alpha\|_\infty \leq 2\beta_{ag}$. Also, $\mu((c - c')(\underline{f}_0 - \underline{f}'_0)) \leq \min(d, \mu(c - c')\mu(\underline{f}_0 - \underline{f}'_0))$. Of course, $\mu(c - c') \leq 2\omega_{ch}$ and $\mu(\underline{f}_0 - \underline{f}'_0) \leq 2\omega_{sk}$. Also, $\min(d, 2\omega_{ag}, \min(d, 4\omega_{ch}\omega_{sk})) = \min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})$. Hence, this first condition becomes

$$\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})(2\beta_{ag})\left\|(c - c')(\underline{f}_0 - \underline{f}'_0)\right\|_\infty < \frac{p-1}{2}$$

Moreover, we have the following.

$$\begin{aligned}\left\|(c - c')(\underline{f}_0 - \underline{f}'_0)\right\|_\infty \leq& \min(d, \mu(c - c'), \mu(\underline{f}_0 - \underline{f}'_0))\|c - c'\|_\infty \left\|\underline{f}_0 - \underline{f}'_0\right\|_\infty\\ \leq& \min(d, 2\omega_{ch}, 2\omega_{sk})4\beta_{ch}\beta_{sk}\end{aligned}$$

Hence, the first condition reduces to $8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} < \frac{p-1}{2}$. Now consider the second condition. We have that $\mu(c - c') \leq 2\omega_{ch}$, but we also have $\mu(\underline{f}_0 - \underline{f}'_0) \leq 2\omega_{sk}$, $\|c - c'\|_\infty \leq 2\beta_{ch}$, and $\left\|\underline{f}_0 - \underline{f}'_0\right\|_\infty \leq 2\beta_{ch}$. Hence, this second condition reduces to $\min(d, 2\omega_{ch}, 2\omega_{sk})4\beta_{ch}\beta_{sk} < \frac{p-1}{2}$. However, $1 \leq 2\min(2\omega_{ag}, 4\omega_{ch}\omega_{sk}\beta_{ag}$, so the first condition implies the second condition.

Since $\widetilde{\mathbb{W}}(\mathcal{V}_{\mathfrak{p}}, \beta, \omega) \subseteq \mathbb{W}(\mathcal{V}_{\mathfrak{p}}, \beta, \omega)$, note that Theorem 5 is sufficient to show there can be only one key that can explain the signature.

Theorem 6. *Let ϵ_{ag}, $\epsilon_{ch}, \epsilon_{\mathcal{A}}, \epsilon' \in (0, 1)$, $t, t_0, t_1, t_2, t' \geq 0$, and $\omega, \beta \in \mathbb{N}$ such that all the following hold.*

$$\begin{aligned}\epsilon_{ag} =& \binom{d}{\omega_{ag}}^{-1} (2\beta_{ag})^{-\omega_{ag}}\\ \epsilon_{ch} =& \binom{d}{\omega_{ch}}^{-1} (2\beta_{ch})^{-\omega_{ch}}\\ t' =& 2(t + t_0) + t_1 + t_2\\ \epsilon' =& \frac{1}{2}(1 - \epsilon_{ch})^2 (1 - \epsilon_{ag})^2 \left(1 - \frac{\epsilon_{ag}}{(1 - \epsilon_{ch})(1 - \epsilon_{ag})\epsilon_{\mathcal{A}}}\right)\epsilon_{\mathcal{A}}^2\\ \omega \geq& \min(d, 2\omega_v + 2\omega_{ag}\omega'_v)\\ \beta \geq& 2\beta_v + 2\min(d, 2\omega_{ag}, \omega'_v)\beta_{ag}\beta'_v\end{aligned}$$

If (i) Π_F is correct and (ii) the hypotheses of Theorems 4 and 5 hold and (iii) there exists some PPT algorithm $\mathcal{A}$ is a $(t, \epsilon_{\mathcal{A}})$-forger of Definition 4 for Π_F which makes at most 1 *query* *to H_{ag} and at most K queries to H_{ch}, then there exists an algorithm $\mathcal{B}$ that is a (t', ϵ')-solver of the $(e, p, \ell, \|\cdot\|_{\infty}, \beta)$-RSIS game of Definition 5 whose solution to that game has Hamming weight at most ω.*

Proof. In the following, we say that query inp made to H_{ag} is *well-formed* if there exists some $1 \leq N \leq K$ such that (i) $\mathsf{inp} = (\mathsf{keys}, \mathsf{msgs}, \mathsf{challs})$, (ii) $\mathsf{keys} = (\mathsf{vk}_i)_{i \in [N]}$ is a sorted tuple of distinct verification keys with $\mathsf{vk}_i \in \mathcal{R}_{\mathsf{p}}^2$ for each $i \in [N]$, (iii) $\mathsf{msgs} = (m_i)_{i \in [N]}$ is a tuple of messages with $m_i \in \{0,1\}^*$ for each $i \in [N]$, and (iv) $\mathsf{challs} = (c_i)_{i \in [N]}$ is a tuple of signature challenges such that $c_i = \mathsf{H}_c(\mathsf{vk}_i, m_i)$ for each $i \in [N]$. Otherwise, we say it is *malformed.* Moreover, we say the query *contains the challenge key* if it is well-formed and there exists some index i^* such that $\mathsf{vk}_{i^*} = \mathsf{vk}^*$.

To prove the theorem, we make the following assumptions on $\mathcal{A}$.

- $\mathcal{A}$ outputs an unsuccessful forgery with probability 0, a successful forgery with probability ϵ, and the failure symbol $\perp$ with probability $1-\epsilon$; $\mathcal{A}$ never outputs an unsuccessful forgery, outputting $\perp$ instead.
- In every transcript producing a successful forgery, say $\mathsf{forg} = ((\mathsf{vk}_i, m_i)_{i \in [N]}, \xi_{ag})$ for some $1 \leq N \leq K$, $\mathcal{A}$ queries H_{ch} with each (vk_i, m_i) except perhaps with probability ϵ_{ch},
- In every transcript producing a successful forgery, $\mathcal{A}$ queries H_{ag} with a well-formed query containing the challenge key at some index i^* except perhaps with probability ϵ_{ag}.

Construction of the RSIS Solver. We construct $\mathcal{B}$ as follows. We wrap $\mathcal{A}$ in a simulation algorithm $\mathcal{S}$ which has the same oracle access to H_{ch} and $\mathcal{O}_{\mathsf{Sign}}$ as $\mathcal{A}$, and simulates the oracle response for the query made to H_{ag}, and is suitable for use in the forking lemma. Next, we construct the forking algorithm $\mathsf{Fork}_{\mathcal{S}}$ to have the same oracle access to H_{ch} and $\mathcal{O}_{\mathsf{Sign}}$ as $\mathcal{S}$, and which provides to $\mathcal{S}$ the responses for the query made to H_{ag} by $\mathcal{A}$. We wrap $\mathsf{Fork}_{\mathcal{S}}$ in $\mathcal{B}$, which simulates the oracle responses for the queries made to H_{ch} and $\mathcal{O}_{\mathsf{Sign}}$ by $\mathsf{Fork}_{\mathcal{S}}$, and which plays the game of Definition 5. This way, $\mathcal{B}$, $\mathsf{Fork}_{\mathcal{S}}$, and $\mathcal{S}$ together play the challenger in the game of 4 against $\mathcal{A}$ to obtain two forgeries.

We now specify these algorithms. We then demonstrate the correctness of $\mathcal{B}$ and compute its run-time and probability of success.

Forger $\mathcal{A}(\tau_{\mathcal{A}}, \lambda, \rho, \mathsf{vk}^*) \to \{\perp, \mathsf{forg}\}$ inputs $(\tau_{\mathcal{A}}, \lambda, \rho, \mathsf{vk}^*)$ for a random tape $\tau_{\mathcal{A}}$, security parameter λ, public parameters ρ for Π_F, and the challenger's public verification key vk^*. Then, $\mathcal{A}$ is granted access to H_{ag} and H_{ch}, and one-time access to $\mathcal{O}_{\mathsf{Sign}}$. In the sequel, denote the query made to $\mathcal{O}_{\mathsf{Sign}}$ with $m_{\mathcal{O}}$, and the response with $\xi_{\mathcal{O}}$. The forger $\mathcal{A}$ makes oracle queries adaptively in any order and eventually outputs a purported forgery forg, or the failure symbol $\perp$. We make no further specifications about how $\mathcal{A}$ works.

Simulation $\mathcal{S}(\tau_{\mathcal{S}}, \lambda, \rho, \mathsf{vk}^*, \underline{\alpha}) \to \{\perp_0, \perp_1, \perp_2, \mathsf{forg}\}$ inputs $(\tau_{\mathcal{S}}, \lambda, \rho, \mathsf{vk}^*, \underline{\alpha}_i)$ for a random tape $\tau_{\mathcal{S}}$, security parameter λ, public parameters ρ for Π_F, the challenger's public verification key vk^*, and aggregation coefficients $\underline{\alpha} = (\underline{\alpha}_i)_{i \in [K]} \in \mathbb{W}^K(\mathcal{R}, \infty, \beta_{ag}, \omega_{ag})$ for simulating a response for a query made to H_{ag}. Then, $\mathcal{S}$ is granted access to H_{ch} and one-time access to $\mathcal{O}_{\mathsf{Sign}}$. The simulation $\mathcal{S}$ runs $\mathcal{A}$ as a sub-routine, using its own oracle access and $\underline{\alpha}$ to play the challenger in the game of Definition 4 as described below. Eventually, $\mathcal{S}$ outputs auxiliary data forg or the failure symbol $\perp$. The simulation $\mathcal{S}$ works as follows.

1. The simulation $\mathcal{S}$ samples $\tau_{\mathcal{A}}$.
2. $\mathcal{S}$ computes $\mathsf{out}_{\mathcal{A}} \leftarrow \mathcal{A}(\tau_{\mathcal{A}}, \lambda, \rho, \mathsf{vk}^*)$, handling oracle queries made by $\mathcal{A}$ as follows.
   - When $\mathcal{A}$ queries H_{ch} or $\mathcal{O}_{\mathsf{Sign}}$, $\mathcal{S}$ answers by using its own oracle access, sending the responses to $\mathcal{A}$ without alteration.
   - When $\mathcal{A}$ makes the query inp to H_{ag}, $\mathcal{S}$ responds with $\underline{\alpha}$ in a way that is dependent on whether inp is well-formed and contains the challenge key.
     - If inp is malformed or if the query is well-formed but the query does not contain the challenge key, respond with $\underline{\alpha}$.

- Otherwise, the query is well-formed and the query contains the challenge key. Since it is a well-formed query, let i^* be the index such that $\mathsf{vk}_{i^*} = \mathsf{vk}^*$. In this case, $\mathcal{S}$ responds by swapping the i^*-th coordinate of $\underline{\alpha}$ with the final coordinate, responding with $\underline{\alpha}^* = (\underline{\alpha}_i^*)_{i \in [K]}$ where $\underline{\alpha}_{i^*}^* = \underline{\alpha}_{K-1}$, where $\underline{\alpha}_{K-1}^* = \underline{\alpha}_{i^*}$, and every other $\underline{\alpha}_i^* = \underline{\alpha}_i$.

3. If $\mathsf{out}_{\mathcal{A}} = \bot$, $\mathcal{S}$ outputs a distinguished failure symbol $\bot_0$ and terminates.
4. Otherwise, $\mathcal{A}$ outputs some $\mathsf{out}_{\mathcal{A}} = \mathsf{forg} = ((\mathsf{vk}_i, m_i)_{i \in [N]}, \xi_{ag})$. In this case, $\mathcal{S}$ looks at its transcript with H_{ch} to find each $c_i = \mathsf{H}_{ch}(\mathsf{vk}_i, m_i)$. If not all of these queries were made, $\mathcal{S}$ outputs $\bot_1$ and terminates.
5. Otherwise, all these queries were made to H_{ch}. In this case, $\mathcal{S}$ looks at its transcript with $\mathcal{A}$ to see if $(\mathsf{vk}_i, m_i, c_i)_{i \in [N]}$ was queried to H_{ag}. If not, then then $\mathcal{S}$ outputs $\bot_2$ and terminates.
6. Otherwise, $\mathcal{S}$ outputs forg.

Forking Algorithm $\mathsf{Fork}_{\mathcal{S}}(\tau_{\mathsf{Fork}_{\mathcal{S}}}, \lambda, \rho, \mathsf{vk}^*) \to \{\bot_0, \bot_1, (\mathsf{aux}_0, \mathsf{aux}_1)\}$ inputs $(\tau_{\mathsf{Fork}_{\mathcal{S}}}, \lambda, \rho, \mathsf{vk}^*)$ for a random tape $\tau_{\mathsf{Fork}_{\mathcal{S}}}$, security parameter λ, public parameters ρ for Π_F, and the challenger's public verification key vk^*. Then $\mathsf{Fork}_{\mathcal{S}}$ is granted access to H_{ch} and one-time access to $\mathcal{O}_{\mathsf{Sign}}$. The forking algorithm $\mathsf{Fork}_{\mathcal{S}}$ runs two instances of $\mathcal{S}$ as sub-routines, as described below, making oracle queries only when $\mathcal{S}$ makes oracle queries, and eventually outputs some data $(\mathsf{aux}_0, \mathsf{aux}_1)$ or one of the distinguished failure symbols $\bot_k$ for some $k = 0, 1$. The forking algorithm works as follows.

1. The forking algorithm $\mathsf{Fork}_{\mathcal{S}}$ samples a random tape τ_S and samples aggregation coefficients $\underline{\alpha}^{(0)} \in \mathbb{W}^K(\mathcal{R}, \infty, \beta_{ag}, \omega_{ag})$.
2. The forking algorithm $\mathsf{Fork}_{\mathcal{S}}$ computes $\mathsf{out}_0 \leftarrow \mathcal{S}(\tau_S, \lambda, \rho, \mathsf{vk}^*, \underline{\alpha}^{(0)})$, handling oracle queries made by $\mathcal{S}$ by using its own oracle access, sending the responses to $\mathcal{S}$ without alteration.
3. If $\mathsf{out}_0 = \bot_k$ for any k, $\mathsf{Fork}_{\mathcal{S}}$ outputs $\bot_0$ and terminates.
4. Otherwise, $\mathsf{out}_0 = \mathsf{forg}_0$ for some $\mathsf{forg}_0 = ((\mathsf{vk}_i^{(0)}, m_i^{(0)})_{i \in [N_0]}, \xi_{ag}^{(0)})$. In this case, $\mathsf{Fork}_{\mathcal{S}}$ sets $\mathsf{aux}_0 = (\mathsf{forg}_0, \underline{\alpha}_{K-1}^{(0)})$.
5. Next, $\mathsf{Fork}_{\mathcal{S}}$ samples $\alpha^{(1)} \in \mathbb{W}(\mathcal{R}_{\mathsf{p}}, \infty, \beta_{ag}, \omega_{ag})$, re-sampling until $\alpha^{(1)} \neq \underline{\alpha}_{K-1}^{(0)}$.
6. Set $\underline{\alpha^{(1)}} = (\underline{\alpha}_0^{(0)}, \underline{\alpha}_1^{(0)}, \ldots, \underline{\alpha}_{K-2}^{(0)}, \alpha^{(1)})$ and compute $\mathsf{out}_1 \leftarrow \mathcal{S}(\tau_S, \lambda, \rho, \mathsf{vk}^*, \underline{\alpha}^{(1)})$, handling oracle queries as follows.
   - When $\mathcal{S}$ queries H_{ch}, $\mathsf{Fork}_{\mathcal{S}}$ uses its own oracle access, sending the response to $\mathcal{S}$ unaltered.
   - When $\mathcal{S}$ queries $\mathcal{O}_{\mathsf{Sign}}$ with some $m_{\mathcal{O}}$, $\mathsf{Fork}_{\mathcal{S}}$ first checks to see if this query matches the query from the first execution of $\mathcal{S}$. If not, $\mathsf{Fork}_{\mathcal{S}}$ outputs $\bot_1$ and terminates. Otherwise, $\mathsf{Fork}_{\mathcal{S}}$ responds with the same signing oracle response $\xi_{\mathcal{O}}$ as in the first execution.
7. If $\mathsf{out}_1 = \bot_k$ for any k, $\mathsf{Fork}_{\mathcal{S}}$ outputs $\bot_1$ and terminates.
8. Otherwise, $\mathsf{out}_1 = \mathsf{forg}_1$ for some $\mathsf{forg}_1 = ((\mathsf{vk}_i^{(1)}, m_i^{(1)})_{i \in [N_1]}, \xi_{ag}^{(1)})$. In this case, $\mathsf{Fork}_{\mathcal{S}}$ sets $\mathsf{aux}_1 = (\mathsf{forg}_1, \alpha^{(1)})$ and outputs $(\mathsf{aux}_0, \mathsf{aux}_1)$.

RSIS Solver $\mathcal{B}(\tau_{\mathcal{B}}, \lambda, \rho_{RSIS}, \underline{a}) \to \{\bot_0, \bot_1, \underline{\xi}^*\}$ inputs $(\tau_{\mathcal{B}}, \lambda, \underline{a})$ for a random tape $\tau_{\mathcal{B}}$, security parameter λ, game parameters ρ_{RSIS}, and the $(e, p, \ell, \|\cdot\|_\infty, \beta)$-RSIS challenge. $\mathcal{B}$ simulates responses to H_{ch} by keeping an internal table T_{ch} and providing a simulated response to $\mathcal{O}_{\mathsf{Sign}}$ for use in the game of Definition 4, and offloading the simulated responses for H_{ag} queries to $\mathcal{S}$. Eventually, $\mathcal{B}$ outputs a solution ξ^* to the game from Definition 5 or a distinguished failure symbol $\bot_0$ or $\bot_1$. $\mathcal{B}$ works as follows.

1. Sample $\tau_{\mathsf{Fork}_{\mathcal{S}}}$ and instantiate a new empty table T_{ch} to keep consistency in simulated responses to H_{ch} oracle queries.

2. Compute the Fusion parameters $\rho \leftarrow \mathsf{Pgen}(\lambda)$, swapping out the public vector for the RSIS challenge $\underline{a}$. Then, using the swapped parameters, sample an honest keypair $(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow \mathsf{KGen}(\rho^*)$ and parse the secret signing key $(\underline{f}_0^*, \underline{f}_1^*) \leftarrow \mathsf{sk}^*$.
3. Compute $\mathsf{out}_{\mathsf{Fork}} \leftarrow \mathsf{Fork}_{\mathcal{S}}(\tau_{\mathsf{Fork}_{\mathcal{S}}}, \lambda, \rho^*, \mathsf{vk}^*)$, handling oracle queries as follows.
   – When $\mathsf{Fork}_{\mathcal{S}}$ makes a query to H_{ch}, say inp, first check if $\mathsf{inp} \in T_{ch}$. If so, respond with $T_{ch}[\mathsf{inp}]$. Otherwise, sample $c \leftarrow\!\!\$\ \mathbb{W}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ch}, \omega_{ch})$, set $T_{ch}[\mathsf{inp}] = c$, and respond with $T_{ch}[\mathsf{inp}]$.
   – When $\mathsf{Fork}_{\mathcal{S}}$ makes its only query to $\mathcal{O}_{\mathsf{Sign}}$, say $m_{\mathcal{O}}$, first check if $(\mathsf{vk}^*, m_{\mathcal{O}}) \in T_{ch}$. If not, simulate $c_{\mathcal{O}} \leftarrow \mathsf{H}_{ch}(\mathsf{vk}^*, m_{\mathcal{O}})$ as described above. Then compute $\xi_{\mathcal{O}} = \underline{f}_0^* c_{\mathcal{O}} + \underline{f}_1^*$ and respond to the query with $\xi_{\mathcal{O}}$.
4. If $\mathsf{out}_{\mathsf{Fork}} = \bot_k$ for any $\bot_k$, output $\bot_0$ and terminate.
5. Otherwise, $\mathsf{out} = (\mathsf{aux}_0, \mathsf{aux}_1)$. Parse the auxiliary data $(\mathsf{forg}_k, \alpha^{(k)}) \leftarrow \mathsf{aux}_k$ for each $k = 0, 1$ and parses the forgeries $((\mathsf{vk}_i^{(k)}, m_i^{(k)})_{i \in [N_0]}, \xi_{ag}^{(k)}) \leftarrow \mathsf{forg}_k$ for each $k = 0, 1$. Since $\mathcal{S}$ did not output $\bot_3$ to get to this point, there exists a pair of indices i_0^*, i_1^* such that $\mathsf{vk}_{i_0^*}^{(0)} = \mathsf{vk}_{i_1^*}^{(1)} = \mathsf{vk}^*$. Since $\mathcal{S}$ did not output $\bot_2$ to get to this point, $(\mathsf{vk}^*, m_{i^*}^{(0)}) \in T_{ch}$. Set $c^* = T_{ch}[(\mathsf{vk}^*, m_{i_0^*}^{(0)})]$, set $\xi' = \xi_{ag}^{(0)} - \xi_{ag}^{(1)}$, and compute
$$\xi^* = \xi' - (\alpha^{(0)} - \alpha^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*).$$
6. If $\xi^* = 0$, output $\bot_1$ and terminate. Otherwise, output ξ^*.

Run-time. Assessing the run-time is simplest, so we begin there. The simulation algorithm $\mathcal{S}$ runs $\mathcal{A}$, which takes time at most t, and runs some additional computations which take additional time, at most some t_0. So the run-time of $\mathcal{S}$ is at most $t + t_0$. The forking algorithm $\mathsf{Fork}_{\mathcal{S}}$ runs two executions of $\mathcal{S}$, each taking time at most $t + t_0$, and runs some additional computations which take at most some time t_1. So $\mathsf{Fork}_{\mathcal{S}}$ takes time at most $2(t + t_0) + t_1$. Lastly, $\mathcal{B}$ runs $\mathsf{Fork}_{\mathcal{S}}$, which takes time at most $2(t+t_0)+t_1$, and some additional computations, which take some time at most t_2. So $\mathcal{B}$ takes time at most $2(t + t_0) + t_1 + t_2$.

Correctness. Demonstrating that $\mathcal{B}$ is correct is the next simplest, so we move onto that. For correctness, we first show that $\mathcal{B}$ correctly plays the game of Definition 5 and correctly simulates the challenger in the unforgeability game of Definition 4 in both executions of $\mathcal{A}$. Note that $\mathcal{B}$ plays the game of Definition 5 by construction. In particular, $\mathcal{B}$ inputs ρ_{RSIS} and $\underline{a}$, and outputs either a failure symbol or a potential solution ξ^*.

Next note that $\mathcal{B}$ correctly simulates the unforgeability game for $\mathcal{A}$. Certainly, begins by sampling keys exactly as in KGen and sending ρ and vk^* to $\mathcal{A}$ as in step (1) of the game from Definition 4. In the course of the execution of $\mathcal{A}$, $\mathcal{B}$ correctly simulates H_{ag} and H_{ch} responses indistinguishably for $\mathcal{A}$. To see why, note that all simulated responses to H_{ch} are independent uniformly random variables with support $\mathbb{W}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ch}, \omega_{ch})$ and the simulated response to the H_{ag} query consists of independent uniformly distributed random variables with support $\mathbb{W}(\mathcal{R}_{\mathfrak{p}}, \infty, \beta_{ag}, \omega_{ag})$. Hence, H_{ch} and H_{ag} are simulated correctly under the random oracle model. Also, in the event $E'(\Pi_F, \lambda)$ from step (2) in the game of Definition 4, $\mathcal{B}$ responds to $\mathcal{O}_{\mathsf{Sign}}$ queries by using the sampled challenge key sk^* according to Sign. So, from the correctness of Π_F, the oracle-generated signature $\xi_{\mathcal{O}}$ satisfies the requisite property for oracle-generated signature from step (2).

Note that Lemma 4 provides that at least two challenge keys could explain the challenge key vk^* and the oracle response $\xi_{\mathcal{O}}$ in the view of $\mathcal{A}$, and that sk^* was sampled uniformly according to KGen. Hence revealing only the sampled public verification key and the oracle-generated signature keeps the challenge key sk^* information-theoretically hidden from $\mathcal{A}$. Moreover, the key sk^* was sampled uniformly and independently, so the view of $\mathcal{A}$ is indistinguishable from the game of Definition 4.

Success probability. Next we show that if $\mathcal{B}$ does not output any $\perp_k$, then $\mathcal{B}$ succeeds at the $(e, p, \ell, \|\cdot\|_\infty, \beta^*)$-RSIS game. To see why ξ^* is a valid solution for the $(e, p, \ell, \|\cdot\|_\infty, \beta^*)$-RSIS game, we first show that ξ^* is in the nullspace of $\langle \underline{a}, - \rangle$, and then we show that $\|\xi^*\|_\infty \leq \beta$ and $\mu(\xi^*) \leq \omega$. Let E_{ξ^*} be the event that $\mathcal{B}$ does not output any $\perp_k$.
First, consider the nullspace of $\langle \underline{a}, - \rangle$. The event that either $\xi_{ag}^{(0)}$ or $\xi_{ag}^{(1)}$ fails verification is a sub-event of the event in which $\mathcal{B}$ outputs $\perp_0$, which is disjoint from E_{ξ^*}. Thus, $\xi_{ag}^{(0)}$ and $\xi_{ag}^{(1)}$ both pass verification. By construction, each α_i except the ones corresponding to the challenge key are equal in both transcripts with probability 1. Also by construction, the coefficients corresponding to the challenge keys, say $\alpha^{(0)}$ and $\alpha^{(1)}$, are not equal in both transcripts with probability 1. The messages m_i, the signature challenges c_i, and the keys vk_i are all included in the query made to H_{ag}, and so were decided upon before the forking point. Thus, they are also all equal in both transcripts with probability 1. We have the following.

$$\begin{aligned}
\langle \underline{a}, \xi_{ag}^{(0)} \rangle &= \sum_{i \in [N]} \alpha_i^{(0)} (\mathsf{vk}_{i,0}^{(0)} c_i^{(0)} + \mathsf{vk}_{i,1}^{(0)}) \text{ and} \\
\langle \underline{a}, \xi_{ag}^{(1)} \rangle &= \sum_{i \in [N]} \alpha_i^{(1)} (\mathsf{vk}_{i,0}^{(1)} c_i^{(1)} + \mathsf{vk}_{i,1}^{(1)}) \text{ so} \\
\langle \underline{a}, \xi_{ag}^{(0)} \rangle - \langle \underline{a}, \xi_{ag}^{(1)} \rangle &= \sum_{i \in [N]} \alpha_i^{(0)} (\mathsf{vk}_{i,0}^{(0)} c_i^{(0)} + \mathsf{vk}_{i,1}^{(0)}) - \sum_{i \in [N]} \alpha_i^{(1)} (\mathsf{vk}_{i,0}^{(1)} c_i^{(1)} + \mathsf{vk}_{i,1}^{(1)}) \\
\langle \underline{a}, \xi' \rangle &= (\alpha^{(0)} - \alpha^{(1)})(\mathsf{vk}_0^* c^* + \mathsf{vk}_1^*)
\end{aligned}$$

Since $\xi^* = \xi' - (\alpha^{(0)} - \alpha^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*)$, this shows ξ^* is in the nullspace of $\langle \underline{a}, - \rangle$. Next, consider the Hamming weight of ξ^*. By correctness, $\mu(\xi_{ag}) \leq \omega_v$. Hence $\mu(\xi') \leq \min(d, 2\omega_v)$. But also, $\mu((\alpha^{(0)} - \alpha^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*) \leq \min(d, \mu(\alpha^{(0)} - \alpha^{(1)})\mu(\underline{f}_0^* c^* + \underline{f}_1^*))$. By correctness, $\mu(\underline{f}_0^* c^* + \underline{f}_1^*) \leq \omega_v'$, and by construction, $\mu(\alpha^{(0)} - \alpha^{(1)}) \leq 2\omega_{ag}$. Hence, $\min(d, \mu(\alpha^{(0)} - \alpha^{(1)})\mu(\underline{f}_0^* c^* + \underline{f}_1^*)) \leq \min(d, 2\omega_{ag}\omega_v')$. Hence, $\mu(\xi^*) \leq \min(d, \min(d, 2\omega_v) + \min(d, 2\omega_{ag}\omega_v')) = \min(d, 2\omega_v + 2\omega_{ag}\omega_v')$. In particular, if $\min(d, 2\omega_v + 2\omega_{ag}\omega_v') \leq \min(d, \omega)$, then ξ^* has Hamming weight bounded by ω.
Next, consider the norm of ξ^*. By correctness, $\|\xi_{ag}\|_\infty \leq \beta_v$. Hence, if

$$2\beta_v + 2\min(d, 2\omega_{ag}, \omega_v')\beta_{ag}\beta_v' \leq \beta$$

then we have the following.

$$\begin{aligned}
\|\xi^*\|_\infty &= \left\| \xi_{ag}^{(0)} - \xi_{ag}^{(1)} - (\alpha_{i^*}^{(0)} - \alpha_{i^*}^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*) \right\|_\infty \\
&\leq \left\| \xi_{ag}^{(0)} \right\|_\infty + \left\| \xi_{ag}^{(1)} \right\|_\infty + \left\| (\alpha_{i^*}^{(0)} - \alpha_{i^*}^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*) \right\|_\infty \\
&\leq 2\beta_v + \left\| (\alpha_{i^*}^{(0)} - \alpha_{i^*}^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*) \right\|_\infty \\
&\leq 2\beta_v + \min\left(d, \mu\left(\alpha_{i^*}^{(0)} - \alpha_{i^*}^{(1)}\right), \mu\left(\underline{f}_0^* c^* + \underline{f}_1^*\right)\right) \left\| \alpha_{i^*}^{(0)} - \alpha_{i^*}^{(1)} \right\|_\infty \left\| \underline{f}_0^* c^* + \underline{f}_1^* \right\|_\infty \\
&\leq 2\beta_v + 2\min(d, 2\omega_{ag}, \omega_v')\beta_{ag}\beta_v' \\
&\leq \beta
\end{aligned}$$

Thus, conditioned upon the event E_{ξ^*}, $\mathcal{B}$ succeeds at the $(e, p, \ell, \|\cdot\|_\infty, \beta)$-RSIS game from Definition 5. In particular, $\mathbb{P}[\mathcal{B} \text{ succeeds}] = \mathbb{P}[E_{\xi^*}]$.
To compute $\mathbb{P}[E_{\xi^*}]$, consider the advantage of each of our specified algorithms in turn, recalling that $\mathcal{A}$ has advantage $\epsilon_{\mathcal{A}}$ by assumption. Note that $\mathcal{S}$ fails if it outputs any $\perp_k$, and each of

these outcomes is a disjoint event. Thus, $\mathbb{P}\left[\vee_{k\in[3]}\bot_k \leftarrow \mathcal{S}\right] = \sum_{k\in[3]} \mathbb{P}\left[\bot_k \leftarrow \mathcal{S}\right]$. Moreover, the event in which $\bot_1 \leftarrow \mathcal{S}$ is a sub-event of the event in which $\bot_0 \not\leftarrow \mathcal{S}$. Also, the event in which $\bot_2 \leftarrow \mathcal{S}$ is a sub-event of the intersection of the events $\bot_0 \not\leftarrow \mathcal{S}$ and $\bot_1 \not\leftarrow \mathcal{S}$.
Note that $\mathcal{S}$ outputs $\bot_1$ if and only if $\mathsf{out}_{\mathcal{A}} = \bot$. By assumption, $\mathcal{A}$ has success probability $\epsilon_{\mathcal{A}}$. In the event that $\bot_0 \not\leftarrow \mathcal{S}$, $\mathcal{S}$ outputs $\bot_1$ if and only if $\mathcal{A}$ did not query H_{ch} for some c_i. By assumption on $\mathcal{A}$, this occurs with probability ϵ_{ch}. Lastly, in the event that $\mathcal{S}$ does not output $\bot_0$ or $\bot_1$, $\mathcal{S}$ outputs $\bot_2$ if and only if $\mathcal{A}$ did not query H_{ag} with the data corresponding to the forgery. In this case, $\mathcal{A}$ must guess an aggregation coefficient, so this occurs with probability ϵ_{ag}. Hence, we have that the probability that $\mathcal{S}$ outputs any $\bot_k$ is at most

$$\epsilon_{ag} + (1-\epsilon_{ag})\left((1-\epsilon_{\mathcal{A}}) + \epsilon_{\mathcal{A}}\left(\epsilon_{ch} + (1-\epsilon_{ch})\epsilon_{ag}\right)\right).$$

Then $\mathcal{S}$ has the following advantage.

$$\begin{aligned}\epsilon_{\mathcal{S}} =& 1 - (1-\epsilon_{\mathcal{A}}) - \epsilon_{\mathcal{A}}\left(\epsilon_{ch} + (1-\epsilon_{ch})\epsilon_{ag}\right)\\ =& \epsilon_{\mathcal{A}}\left(1-\epsilon_{ch} - (1-\epsilon_{ch})\epsilon_{ag}\right)\\ =& \epsilon_{\mathcal{A}}\left(1-\epsilon_{ch}\right)\left(1-\epsilon_{ag}\right)\end{aligned}$$

Next consider the advantage of $\mathsf{Fork}_{\mathcal{S}}$, say ϵ_{Fork}. The General Forking Lemma claims that the advantage of $\mathsf{Fork}_{\mathcal{S}}$ satisfies $\epsilon_{\mathsf{Fork}} \geq \epsilon_{\mathcal{S}}\left(\epsilon_{\mathcal{S}} - \epsilon_{ag}\right)$. Lastly, consider the advantage of $\mathcal{B}$, say $\epsilon_{\mathcal{B}}$, which is the probability that $\mathcal{B}$ does not output any $\bot_0$ or $\bot_1$. Note that, again, each of these is a disjoint event. Thus, $\mathbb{P}\left[\vee_{k=0}^{1}\bot_k \leftarrow \mathcal{B}\right] = \sum_{k=0}^{1} \mathbb{P}\left[\bot_k \leftarrow \mathcal{B}\right]$. Similarly to before, we have that the event in which $\bot_1 \leftarrow \mathcal{B}$ is a sub-event of the event in which $\bot_0 \not\leftarrow \mathcal{B}$. Note that $\mathcal{B}$ outputs $\bot_0$ if and only if $\mathsf{Fork}_{\mathcal{S}}$ fails, which occurs with probability $1-\epsilon_{\mathsf{Fork}}$. Also, in the event that $\bot_0 \not\leftarrow \mathcal{B}$, $\mathcal{B}$ outputs $\bot_1$ if and only if $\xi^* = 0$. This is the case that $\xi' = (\alpha^{(0)} - \alpha^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*)$.
In this case, by Lemma 4, there exists at least two keypairs that can explain the adversary's view before they output a forgery. The challenge key was sampled uniformly and independently at the start of the experiment, so the keypair responsible for the view is kept information-theoretically hidden from the adversary. According to Lemma 5, only one keypair can explain the adversary's view once ξ' is computed. Hence, $\xi' = (\alpha^{(0)} - \alpha^{(1)})(\underline{f}_0^* c^* + \underline{f}_1^*)$ occurs with probability at most 1/2.
Hence, we have that the probability that $\mathcal{B}$ outputs any $\bot_k$ is at most $1-\epsilon_{\mathsf{Fork}}+\frac{\epsilon_{\mathsf{Fork}}}{2} = 1-\frac{\epsilon_{\mathsf{Fork}}}{2}$. The success probability of $\mathcal{B}$ is then $\frac{\epsilon_{\mathsf{Fork}}}{2}$. Combining these results, we have the following as claimed.

$$\begin{aligned}\epsilon_{\mathcal{B}} \geq& \frac{\epsilon_{\mathsf{Fork}}}{2}\\ \geq& \frac{\epsilon_{\mathcal{S}}}{2}\left(\epsilon_{\mathcal{S}} - \epsilon_{ag}\right)\\ \geq& \frac{1}{2}(1-\epsilon_{ch})^2(1-\epsilon_{ag})^2\left(1 - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})\epsilon_{\mathcal{A}}}\right)\epsilon_{\mathcal{A}}^2\end{aligned}$$

Lemma 4. *Let $\lambda \in \mathbb{N}$, $\kappa \in \mathbb{N}_0$ and assume forging a signature from Π_F has λ bits of hardness. If all of the following conditions hold then the proof of Theorem 6 implies a tightness gap $\log_2(\gamma) \leq 9 + \lambda$.*

- $t_0 < t$,
- $t_1 + t_2 < 2(t + t_0)$,
- $\epsilon_{ch} < 2^{-\lambda}$,
- $\epsilon_{ag} < \frac{1}{2}$,
- $\frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \leq 2^{-(\lambda+1)}$.

Proof. We have the following.

$$\begin{aligned}\log_2 \gamma =& \log_2 \left(\frac{2(t+t_0)+t_1+t_2}{t} \frac{2\epsilon_{\mathcal{A}}}{(1-\epsilon_{ch})^2 (1-\epsilon_{ag})^2 \epsilon_{\mathcal{A}} \left(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})}\right)} \right) \\ =& 1 + \log_2 \left(2(1+\frac{t_0}{t}) + \frac{t_1+t_2}{t} \right) - \log_2 \left((1-\epsilon_{ag})^2 (1-\epsilon_{ch})^2 \left(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \right) \right)\end{aligned}$$

Consider the second term. We have that $\log_2(1+\frac{t_0}{t}) < 1$ and $\log_2(1+\frac{t_1+t_2}{2(t+t_0)}) < 1$ by assumption. Thus, we have the following.

$$\begin{aligned}\log_2(2(1+\frac{t_0}{t}) + \frac{t_1+t_2}{t}) =& \log_2 \left(2(1+\frac{t_0}{t}) \left(1 + \frac{t_1+t_2}{2(t+t_0)} \right) \right) \\ =& 1 + \log_2(1+\frac{t_0}{t}) + \log_2 \left(1 + \frac{t_1+t_2}{2(t+t_0)} \right) \leq 3\end{aligned}$$

Substituting this result into the bound on $\log_2(\gamma)$, we have the following.

$$\begin{aligned}\log_2(\gamma) \leq& 4 - \log_2 \left((1-\epsilon_{ag})^2 (1-\epsilon_{ch})^2 \left(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \right) \right) \\ =& 4 - 2\log_2(1-\epsilon_{ag}) - 2\log_2(1-\epsilon_{ch}) - \log_2 \left(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \right)\end{aligned}$$

By assumption, we have the following. First, $\epsilon_{ag} < \frac{1}{2}$, so $-2\log_2(1-\epsilon_{ag}) < 2$. Second, $\epsilon_{ch} < 2^{-\lambda}$, so $-2\log_2(1-\epsilon_{ch}) < -2\log_2(1-2^{-\lambda}) \leq 2$ since $\lambda \geq 1$. Also, $\frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \leq 2^{-(\lambda+1)}$, so $\epsilon_{\mathcal{A}} - 2^{-(\lambda+1)} \leq \epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})}$ and therefore $-\log(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})}) \leq -\log(\epsilon_{\mathcal{A}} - 2^{-(\lambda+1)})$.

$$\begin{aligned}\log_2(\gamma) \leq& 4 - 2\log_2(1-\epsilon_{ag}) - 2\log_2(1-\epsilon_{ch}) - \log_2 \left(\epsilon_{\mathcal{A}} - \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} \right) \\ \leq& 8 - \log(\epsilon_{\mathcal{A}} - 2^{-(\lambda+1)})\end{aligned}$$

In particular, if $\epsilon_{\mathcal{A}} \geq 2^{-\lambda}$, then $\epsilon_{\mathcal{A}} - 2^{-(\lambda+1)} \geq 2^{-\lambda} - 2^{-(\lambda+1)} = 2^{-(\lambda+1)}$ so $\log(\epsilon_{\mathcal{A}} - 2^{-(\lambda+1)}) \geq -(\lambda+1)$. In this case, $\log(\gamma) \leq 9 + \lambda$.

B Parameter Selection Search Heuristic

Consider the existence of solutions to the system of inequalities (1) through (9) from section B. Recall that the Fusion per-signer space complexity is $2d\log(p) + \frac{K\ell}{d}\log(2\beta_v + 1)$.
For any fixed (λ, d), and for any choice of $1 \leq \omega_{ch} \leq d$, the value ϵ_{ch} as a function of β_{ch} is strictly positive, strictly decreasing, and has $\lim_{\beta_{ch}\to\infty} \epsilon_{ch} = 0$. Hence, there exists a critical function of ω_{ch} which depends on the fixed λ and d, say $\beta^*_{ch}(\omega_{ch})$, such that (7) is satisfied by $(\lambda, d, \omega_{ch}, \beta^*_{ch}(\omega_{ch}))$. This still leaves us freedom of choice in selecting ω_{ch}, and we can select ω_{ch} by minimizing per-signer space complexity. The highest order term in the per-signer space complexity that depends on β_{ch} and ω_{ch} is $O(2\log(\omega_{ch}) + \log(\beta_{ch}))$, so we set $\beta_{ch} = \beta^*_{ch}$ and select ω_{ch} to reduce $\log(2\omega_{ch}) + \log(\beta^*_{ch})$.

Next, for these fixed λ, d, ω_{ch}, and β_{ch}, and for any $1 \leq \omega_{ag} \leq d$, the values ϵ_{ag} and $\frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})}$ as functions of β_{ag} are strictly positive, strictly decreasing, and $\lim_{\beta_{ag}\to\infty} \epsilon_{ag} = \lim_{\beta_{ag}\to\infty} \frac{\epsilon_{ag}}{(1-\epsilon_{ch})(1-\epsilon_{ag})} = 0$. Hence, there exists a critical function of ω_{ag} which depends on the fixed $\lambda, d, \omega_{ch}, \beta_{ch}$, say $\beta^*_{ag}(\omega_{ag})$ such that (8) and (9) are both satisfied for any $\beta_{ag} \geq \beta^*_{ag}(\omega_{ag})$. Again, this leaves us freedom of choice in selecting ω_{ag}, and again we can select ω_{ag} by minimizing per-signer space complexity. The highest order term in the per-signer space complexity that depends on ω_{ag} and β_{ag} is $O(\log(\omega_{ag}) + \log(\beta_{ag}))$. We set $\beta_{ag} = \beta^*_{ag}(\lambda, d, \omega_{ch}, \beta_{ch}, \omega_{ag})$ and select ω_{ag} to reduce $\log(\omega_{ag}) + \log(\beta_{ag})$ as before.

Next, for these fixed parameters λ, d, ω_{ch}, β_{ch}, ω_{ag}, β_{ag}, and for some K, there is a subset of parameter space for choices of $(\omega_{sk}, \beta_{sk})$ satisfying (5). Pairs in this subset can be found via brute-force search. From each of these, a bound $p^*(\omega_{sk}, \beta_{sk})$ exists such that (2) and (3) are satisfied for every $p \geq p^*$. Now, since (5) is satisfied, there always exists a minimal $\ell^*(\omega_{sk}, \beta_{sk}, p)$ such that (6) are both satisfied for every $\ell \geq \ell^*$. Moreover, there always exists a critical $\ell^{**}(\omega_{sk}, \beta_{sk}, p)$ such that (4) is satisfied for all $\ell \geq \ell^{**}(\omega_{sk}, \beta_{sk}, p)$. Thus, for any $\ell \geq \max(\ell^*, \ell^{**})$, selecting $(\omega_{sk}, \beta_{sk}, p^*, \ell)$ will provide valid parameters if (4) is satisfied. We can set $\ell = \max(\ell^*, \ell^{**})$, but we still have freedom of choice of $(\omega_{sk}, \beta_{sk})$. As before, we can select $(\omega_{sk}, \beta_{sk})$ to minimize per-signer space complextiy. The highest order term in per-signer space complexity that depends on $(\omega_{sk}, \beta_{sk}, p, \ell)$ is $O\left(2\log(p) + \frac{\ell}{K}\left(\log(\omega_{sk}) + \log(\beta_{sk}) + \log(1 + \omega_{sk}\beta_{ch})\right)\right)$, so we minimize this over our sampled choices.

These facts lend themselves to a heuristic search which works loosely as follows.

- Input λ, d, K.
- Set $1 \leq \omega_{ch} \leq d$ to minimize $2\log(\omega_{ch}) + \log(\beta^*_{ch})$.
- Set $1 \leq \omega_{ag} \leq d$ to minimize $\log(\omega_{ag}) + \log(\beta^*_{ag})$.
- Find a set $\mathcal{S}$ of pairs $(\omega_{sk}, \beta_{sk})$ that satisfy (5).
- For each pair $(\omega_{sk}, \beta_{sk}) \in \mathcal{S}$, compute p^* and $\max(\ell^*, \ell^{**})$.
- Minimize $2\log(p^*) + \frac{\max(\ell^*, \ell^{**})}{K}(\log(\omega_{sk}) + \log(\beta_{sk}) + \log(1 + \omega_{sk}\beta_{ch}))$ to pick $(\omega_{sk}, \beta_{sk})$.

This still provides us a free choice of λ, d, K. We restricted our attention to $\lambda \in \{128, 256, 512\}$ as usual. We restricted our attention to $d \in \{64, 128, 256\}$ because larger d led to key sizes that could not beat CRYSTALS-Dilithium or Falcon. For each choice of λ, d, we then treated K as an input variable to the above heuristic algorithm, whose output was $(\omega_{ch}, \beta_{ch}, \omega_{ag}, \beta_{ag}, \omega_{sk}, \beta_{sk}, p, \ell)$. From this output, we could easily compute the per-signer space complexity of the resulting scheme and the critical number of signatures required to beat CRYSTALS-Dilithium or Falcon.

We first sought the smallest K such that beating CRYSTALS-Dilithium or Falcon with aggregation was possible. We next sought the largest K such that the heuristic algorithm output valid parameters. We then were able to look between this minimum and maximum to seek the choice of K that led to parameters $(\omega_{ch}, \beta_{ch}, \omega_{ag}, \beta_{ag}, \omega_{sk}, \beta_{sk}, p, \ell)$ that led to optimal space-efficiency both with respect to per-signer space complexity and the critical number of signatures required to beat CRYSTALS-Dilithium or Falcon.

C Validity of Parameterization

C.1 Parameterizing Fusion Light

We found that, for $\lambda = 128$, $d = 64$ appears to offer enough security to parameterize the scheme with better space efficiency than Falcon, but not for $d < 64$ or $d > 64$. For these parameters, we have $\beta = 1073231744$ and $p = 2147465729$. So we have $\beta = 1073231744 < 1073732864 = \frac{p-1}{2}$, resolving (1). Also, we have

$$8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} = 8626176 < 1073732864 = \frac{p-1}{2}$$

resolving (2). For security in the underlying RSIS game, we have both

$$(\ell d - 1)^{-1}\left(\frac{\log(\ell)}{2} + \frac{\log(d)}{2} + \log(\beta) - \frac{\log(p)}{\ell}\right) \approx 0.0029364536698398697 \text{ and}$$
$$\frac{\left(\log\left(2\lambda+9\right) + \frac{0.265}{2\lambda+9}\log\left(\pi\cdot\frac{2\lambda+9}{0.265}\right) - \log\left(2\pi e(0.265)\right)\right)}{\left(2\left(\frac{2\lambda+9}{0.265}-1\right)\right)} \approx 0.0029445497471045747$$

resolving (4). Since

$$\lambda + 2d\log(p) + \ell d\log(2\beta'_v + 1) \approx 167061.84629641057 \text{ and}$$
$$2\ell\left(\log\begin{pmatrix} d \\ \omega_{sk}\end{pmatrix} + \omega_{sk}\log(2\beta_{sk})\right) \approx 167242.97536480168$$

we satisfy (6) (and an ℓ satisfying (6) like this implies (5) is also satisfied). Since $\log\binom{d}{\omega_{ch}} + \omega_{ch}\log(2\beta_{ch}) \approx -129.3485086350302 > -128$, we satisfy (7). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -130.26856356544508 < -1$, we satisfy (8). Lastly, since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) - \log(1 - \epsilon_{ch}) - \log(1-\epsilon_{ag}) \approx -\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -130.26856356544508 < -129 = -(\lambda+1)$, we satisfy (9).
For these parameters, we have $\beta'_v = 4264$, $\beta_v = 536070080$, verification keys of 0.496 KB. Signatures take up 21.84 KB, but $K = 1796$ of these combine into a single aggregate signature, which takes up 46.8 KB. This averages to a weight of 0.523 KB per signer. This is more space-efficient than CRYSTALS-Dilithium as long as at least 25 signatures are being aggregated together before verification. This is more space-efficient than Falcon as long as at least 276 signatures are being aggregated together before verification.

C.2 Parameterizing Fusion Mid

We present two parameterizations in Table 4 for Fusion Mid at two different security levels. Users can aggregate a sufficient number of signatures to beat the efficiency of CRYSTALS-Dilithium, but the signatures are not efficient enough to beat Falcon. Fusion Mid users of all security levels use the same ring, gaining security by varying only a few parameters: increase ℓ, ω_{ch}, ω_{ag}, and β_{sk} and decrease aggregation capacity K to increase security. We now show that the Fusion Mid parameter sets satisfy all the requisite conditions.

For $\boldsymbol{\lambda = 128}$ When $\lambda = 128$ with Fusion Mid parameters, we have $\beta = 1073720960$ and $p = 2147465729$. So we have $\beta = 1073720960 < 1073732864 = \frac{p-1}{2}$, resolving (1). Also, we have

$$8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} = 799552 < 1073732864 = \frac{p-1}{2}$$

resolving (2). For security in the underlying RSIS game, we have both

$$(\ell d - 1)^{-1}\left(\frac{\log(\ell)}{2} + \frac{\log(d)}{2} + \log(\beta) - \frac{\log(p)}{\ell}\right) \approx 0.0029384084480779636$$
$$\frac{\left(\log\left(2\lambda+9\right) + \frac{0.265}{2\lambda+9}\log\left(\pi\cdot\frac{2\lambda+9}{0.265}\right) - \log\left(2\pi e(0.265)\right)\right)}{\left(2\left(\frac{2\lambda+9}{0.265}-1\right)\right)} \approx 0.0029445497471045747$$

which resolves (4). Since

$$\lambda + 2d\log(p) + \ell d\log(2\beta'_v + 1) \approx 140931.41794990833 \text{ and}$$
$$2\ell\left(\log\begin{pmatrix} d \\ \omega_{sk}\end{pmatrix} + \omega_{sk}\log(2\beta_{sk})\right) \approx 141553.31908087962$$

we satisfy (6) (and an ℓ satisfying (6) like this implies (5) is also satisfied). Since $\log\binom{d}{\omega_{ch}} + \omega_{ch}\log(2\beta_{ch}) \approx -129.62138779517142 > -128$, we satisfy (7). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -129.62138779517142 < -1$, we satisfy (8). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) - \log(1-\epsilon_{ch}) - \log(1-\epsilon_{ag}) \approx -\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -129.62138779517142 < -129 = -(\lambda+1)$, we satisfy (9).
For these parameters, we have $\beta'_v = 832$, $\beta_v = 536808896$, verification keys of 0.992 KB. Signatures take up 17.072 KB, but $K = 20813$ of these combine into a single aggregate signature, which takes up 46.56 KB. This averages to a weight of 0.995 KB per signer. This is more space-efficient than CRYSTALS-Dilithium as long as at least 33 signatures are being aggregated together before verification.

For $\boldsymbol{\lambda = 256}$ When $\lambda = 256$ with Fusion Mid, we have $\beta = 1071168000$ and $p = 2147465729$. So we have $\beta = 1071168000 < 1073732864 = \frac{p-1}{2}$, resolving (1). Also, we have

$$8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} = 68382720 < 1073732864 = \frac{p-1}{2}$$

resolving (2). For security in the underlying RSIS game, we have both

$$(\ell d - 1)^{-1}\left(\frac{\log(\ell)}{2} + \frac{\log(d)}{2} + \log(\beta) - \frac{\log(p)}{\ell}\right) \approx 0.0017412957321820508$$
$$\frac{\left(\log(2\lambda+9) + \frac{0.265}{2\lambda+9}\log\left(\pi\cdot\frac{2\lambda+9}{0.265}\right) - \log(2\pi e(0.265))\right)}{\left(2\left(\frac{2\lambda+9}{0.265} - 1\right)\right)} \approx 0.0017438059974185121$$

which resolves (4). Since

$$\lambda + 2d\log(p) + \ell d\log(2\beta'_v + 1) \approx 327681.52615705435 \text{ and}$$
$$2\ell\left(\log\begin{pmatrix} d \\ \omega_{sk}\end{pmatrix} + \omega_{sk}\log(2\beta_{sk})\right) \approx 327824.57751873956$$

we satisfy (6) (and an ℓ satisfying (6) like this implies (5) is also satisfied). Since $\log\binom{d}{\omega_{ch}} + \omega_{ch}\log(2\beta_{ch}) \approx -258.4547769280831 > -256$, we satisfy (7). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -257.9700593112159 < -1$, we satisfy (8). Lastly, since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) - \log(1-\epsilon_{ch}) - \log(1-\epsilon_{ag}) \approx -\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -257.9700593112159 < -257 = -(\lambda+1)$, we satisfy (9).
For these parameters, we have $\beta'_v = 16800$, $\beta_v = 531283200$, verification keys of 0.992 KB. Signatures take up 42.496 KB, but $K = 236$ of these combine into a single aggregate signature, which takes up 79.68 KB. This averages to a weight of 1.33 KB per signer. This is more space-efficient than CRYSTALS-Dilithium as long as at least 56 signatures are being aggregated together before verification.

C.3 Parameterizing Fusion Heavy

As with Fusion Mid, we present two parameterizations in Table 4 for Fusion Heavy at two different security levels. As before, users can aggregate a sufficient number of signatures to beat the efficiency of CRYSTALS-Dilithium, but the signatures are not efficient enough to beat Falcon. As with Fusion Mid, Fusion Heavy users of all security levels use the same ring, gaining security by varying only a few parameters: increase ℓ, ω_{ch}, ω_{ag}, and β_{sk} and decrease aggregation capacity K to increase security.
We next show that the Fusion Heavy parameter sets satisfy all the requisite conditions.

For $\boldsymbol{\lambda = 128}$ When $\lambda = 128$, Fusion Heavy has $\beta = 1073717280$ and $p = 2147465729$. So we have $\beta = 1073717280 < 1073732864 = \frac{p-1}{2}$, resolving (1). Also, we have

$$8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} = 507840 < 1073732864 = \frac{p-1}{2}$$

resolving (2). For security in the underlying RSIS game, we have both

$$(\ell d - 1)^{-1}\left(\frac{\log(\ell)}{2} + \frac{\log(d)}{2} + \log(\beta) - \frac{\log(p)}{\ell}\right) \approx 0.00294185848837447794$$

$$\frac{\left(\log(2\lambda+9) + \frac{0.265}{2\lambda+9}\log\left(\pi \cdot \frac{2\lambda+9}{0.265}\right) - \log(2\pi e(0.265))\right)}{\left(2\left(\frac{2\lambda+9}{0.265} - 1\right)\right)} \approx 0.0029445497471045747$$

which resolves (4). Since

$$\lambda + 2d\log(p) + \ell d \log(2\beta_v' + 1) \approx 144936.19140916737 \text{ and}$$

$$2\ell\left(\log\binom{d}{\omega_{sk}} + \omega_{sk}\log(2\beta_{sk})\right) \approx 145167.74327767495$$

we satisfy (6) (and an ℓ satisfying (6) like this implies (5) is also satisfied). Since $\log\binom{d}{\omega_{ch}} + \omega_{ch}\log(2\beta_{ch}) \approx -131.07780800641922 > -128$, we satisfy (7). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -131.07780800641922 < -1$, we satisfy (8). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) - \log(1-\epsilon_{ch}) - \log(1-\epsilon_{ag}) \approx -\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -131.07780800641922 < -129 = -(\lambda+1)$, we satisfy (9).
For these parameters, we have $\beta_v' = 720$, $\beta_v = 536825520$, verification keys of 1.984 KB. Signatures take up 16.896 KB, but $K = 32417$ of these combine into a single aggregate signature, which takes up only 46.08 KB. This averages to a weight of 1.986 KB per signer. This is more space-efficient than CRYSTALS-Dilithium as long as at least 106 signatures are aggregated together before verification.

For $\boldsymbol{\lambda = 256}$ On the other hand, when $\lambda = 256$, we have $\beta = 1073404800$ and $p = 2147465729$. So we have $\beta = 1073404800 < 1073732864 = \frac{p-1}{2}$, resolving (1). Also, we have

$$8\min(d, 2\omega_{ag}, 4\omega_{ch}\omega_{sk})\min(d, 2\omega_{ch}, 2\omega_{sk})\beta_{ag}\beta_{ch}\beta_{sk} = 5990400 < 1073732864 = \frac{p-1}{2}$$

resolving (2). For security in the underlying RSIS game, we have both

$$(\ell d - 1)^{-1}\left(\frac{\log(\ell)}{2} + \frac{\log(d)}{2} + \log(\beta) - \frac{\log(p)}{\ell}\right) \approx 0.0017326480436488268$$

$$\frac{\left(\log(2\lambda+9) + \frac{0.265}{2\lambda+9}\log\left(\pi \cdot \frac{2\lambda+9}{0.265}\right) - \log(2\pi e(0.265))\right)}{\left(2\left(\frac{2\lambda+9}{0.265} - 1\right)\right)} \approx 0.0017438059974185121$$

This resolves (4). Since

$$\begin{aligned}\lambda + 2d\log(p) + \ell d\log(2\beta_v' + 1) &\approx 284520.07556304254 \text{ and}\\ 2\ell\left(\log\binom{d}{\omega_{sk}} + \omega_{sk}\log(2\beta_{sk})\right) &\approx 284741.8862621239\end{aligned}$$

we satisfy (6) (and an ℓ satisfying (6) like this implies (5) is also satisfied). Since $\log\binom{d}{\omega_{ch}} + \omega_{ch}\log(2\beta_{ch}) \approx -257.0147390445861 > -256$, we satisfy (7). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -257.0147390445861 < -1$, we satisfy (8). Since $-\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) - \log(1-\epsilon_{ch}) - \log(1-\epsilon_{ag}) \approx -\log\binom{d}{\omega_{ag}} - \omega_{ag}\log(2\beta_{ag}) \approx -257.0147390445861 < -257 = -(\lambda+1)$, we satisfy (9).

For these parameters, we have $\beta_v' = 3172$, $\beta_v = 536321760$, verification keys of 1.984 KB. Signatures take up 34.528 KB, but $K = 2818$ of these combine into a single aggregate signature, which takes up 79.68 KB. This averages to a weight of 2.013 KB per signer. This is more space-efficient than CRYSTALS-Dilithium as long as at least 183 signatures are being aggregated together before verification.

References

1. Dan Boneh and Sam Kim. One-time and interactive aggregate signatures from lattices. *preprint*, 2020.
2. Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *Theory of Cryptography Conference*, pages 37–54. Springer, 2008.
3. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.
4. Nabil Alkeilani Alkadri, Johannes A Buchmann, Rachid El Bansarkhani, and Juliane Krämer. A framework to select parameters for lattice-based cryptography. *IACR Cryptol. ePrint Arch.*, 2017:615, 2017.
5. Rachel Player. *Parameter selection in lattice-based cryptography.* PhD thesis, Royal Holloway, University of London, 2018.
6. Muhammed Fethullah Esgin. *Practice-Oriented Techniques in Lattice-Based Cryptography.* PhD thesis, Monash University, 2020.