Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium

Manuel Barbosa[1], Gilles Barthe[2], Christian Doczkal[2], Jelle Don[3], Serge Fehr[3,4], Benjamin Grégoire[5], Yu-Hsuan Huang[3], Andreas Hülsing[6], Yi Lee[2,7], and Xiaodi Wu[7]

[1] University of Porto (FCUP) and INESC TEC, Portugal
[2] Max Planck Institute for Security and Privacy, Germany
[3] Centrum Wiskunde & Informatica, The Netherlands
[4] Leiden University, The Netherlands
[5] Inria Centre at Université Côte d'Azur
[6] Eindhoven University of Technology
[7] University of Maryland, United States

`mbb@fc.up.pt`, `{gilles.barthe, christian.doczkal}@mpi-sp.org`, `{jelle.don, serge.fehr, yhh}@cwi.nl`, `benjamin.gregoire@inria.fr`, `andreas@huelsing.net`, `{ylee1228, xiaodiwu}@umd.edu`

Abstract. We extend and consolidate the security justification for the Dilithium signature scheme. In particular, we identify a subtle but crucial gap that appears in several ROM and QROM security proofs for signature schemes that are based on the Fiat-Shamir with aborts paradigm, including Dilithium. The gap lies in the CMA-to-NMA reduction and was uncovered when trying to formalize a variant of the QROM security proof by Kiltz, Lyubashevsky, and Schaffner (Eurocrypt 2018). The gap was confirmed by the authors, and there seems to be no simple patch for it. We provide new, fixed proofs for the affected CMA-to-NMA reduction, both for the ROM and the QROM, and we perform a concrete security analysis for the case of Dilithium to show that the claimed security level is still valid after addressing the gap. Furthermore, we offer a fully mechanized ROM proof for the CMA-security of Dilithium in the EasyCrypt proof assistant. Our formalization includes several new tools and techniques of independent interest for future formal verification results.

1 Introduction

Modern cryptographic standards, including AES and SHA3, are often selected through open, multi-year cryptographic competitions. An important goal of these competitions is to increase confidence in the schemes selected for standardization. To this end, candidate schemes are exposed to scrutiny by the cryptography community. This scrutiny generally yields a combination of cryptanalytic attacks

Authors are listed in alphabetical order; see https://www.ams.org/profession/leaders/culture/JointResearchandItsPublicationfinal.pdf.

and provable security claims. The former leads to schemes being abandoned, narrowing the choice of candidates, while the latter plays a fundamental role in the selection of the remaining candidates. Overall, competitions increase confidence in selected standards. However, competitions are not infallible. In particular, flaws in candidate designs may go undetected by public scrutiny far into the standardization process. These "near misses" beg for complementary methods for validating provable security claims of widely used standards.

Post-Quantum Cryptography and Dilithium. In 2016, NIST initiated a competition for standardizing cryptographic algorithms that could withstand quantum adversaries. The competition recently reached an important milestone with the selection of four standards: one KEM (Kyber) and three signature algorithms (Dilithium, Falcon, SPHINCS+). These algorithms were chosen out of 69 candidates, some of which may still be selected during a fourth round. The selected candidates will form the backbone of quantum-resistant cryptography. Given the stakes, there is ample motivation for supporting all the selected candidates with computer-aided security proofs.

Dilithium [DKL$^+$21,DKL$^+$18] is a lattice-based digital signature based on the Fiat-Shamir with aborts (FSwA) paradigm introduced by Lyubashevsky [Lyu09, Lyu12]. Recall that the classic Fiat-Shamir (FS) paradigm transforms an interactive identification scheme (IDS) based on the standard commit-challenge-response structure into a digital signature scheme. The FS transform takes an IDS scheme ID and a hash function H (which is typically modelled as a random oracle) and sets the signature key pair to be that of ID. Then, to produce a signature on message m, the signer generates a first message w, locally sets the challenge to be $c := \mathsf{H}(w, m)$ and completes the signature as $\sigma := (w, z)$, where z is the response generated by ID upon first message w and challenge c. A signature $\sigma = (w, z)$ is valid if $(w, \mathsf{H}(w, m), z)$ is accepted by ID. The Fiat-Shamir with aborts (FSwA) paradigm extends the FS transform to allow for the response generation procedure to abort[1] — hence FS with aborts — which means that the signing algorithm must now execute the IDS repeatedly until a valid trace (w, c, z) is produced. We will denote this transformation by $\mathsf{FSwA}[\mathsf{ID}, \mathsf{H}]$.

The security of FSwA has been analyzed many times. In particular, the original analysis in [Lyu12] (in the ROM) concludes that the resulting signature scheme is secure down to the underlying lattice-based assumption. Later, Kiltz, Lyubashevsky, and Schaffner [KLS18] (KLS) developed a modular framework that follows the structure of the FSwA transform and used it to extend the results of the security analysis to quantum attackers in the Quantum-accessible Random Oracle Model (QROM).

Computer-Aided Cryptography (CAC). CAC is an emerging approach that develops computer tools for building and independently verifying provable security claims [BBB$^+$21]. CAC formal verification tools have been used to validate the security claims for a number of cryptographic primitives and protocols,

[1] This is necessary for a large class of lattice-based IDS, to avoid leaking the secret key via biased responses z.

and they have progressed to a point where they can be used to increase the level of assurance in standardisation processes. The most outstanding application of CAC to date is arguably the TLS (Transport Layer Security) protocol: the most recent version, TLS 1.3, was designed under the coordination of the IETF with the active involvement of formal verification experts, who used formal tools to unveil logical flaws in previous versions of TLS and intermediate designs, and to validate the security arguments [BBK17, DFK+17, CHSvdM16, CHH+17].

In this paper we focus on EasyCrypt, a tool designed for machine-checking code-based computational security proofs, and hence ideally suited for formally verifying the security proofs for low-level primitives such as digital signature and encryption schemes. EasyCrypt permits stating and proving computational security goals using the same formalisms adopted in cryptographic papers. We report the results of our efforts to formally verify the security proof for the Dilithium signature scheme and provide further evidence that computer-aided cryptography permits guraranteeing the absence of design flaws in cryptographic standards to a much higher level of assurance than manual inspection.

Main Contributions. The main contributions of this paper are three-fold. First, we identify a subtle but crucial gap that appears in several ROM and QROM security proofs of Dilithium and other schemes based on FSwA, including [Lyu12] and [KLS18]. This gap was uncovered when formalizing a variant of the proof in [KLS18]. Second, we provide fixed proofs, both for the ROM and the QROM. Third, we fully mechanize the ROM proof in the EasyCrypt proof assistant. Our formalization includes several new tools and techniques of independent interest for future formal verification results.

We elaborate on these contributions below, but stress at this point two important take-aways: 1) our results extend and consolidate the security justification for the Dilithium signature scheme and 2) the gap in the proof would have been found earlier if any of the affected works, most prominently the Dilithium submission to the NIST post-quantum competition, had been subject to formal verification in the past.

The Gap. The gap in the proof of FSwA occurs in the reduction from chosen message attacks (CMA) to no-message attacks (NMA). In this step, signature queries made by the considered CMA-attacker $\mathcal{A}^{\mathsf{Sign},H}$, which has access to a singing oracle and the random oracle, must be answered without knowledge of the secret key, replacing real signatures with fake ones produced by an Honest-Verifier Zero Knowledge (HVZK) simulator associated with the IDS. To ensure that the attacker cannot detect that it is being given fake signatures, it is also necessary to reprogram the random oracle to be consistent with the transcripts produced by the simulator. The crucial step boils down to replacing the oracle Sign by the oracle Trans (see Fig. 1 below), where Resp is an algorithm that may return $\perp$.

```
Sign(m):
1: repeat
2:     (w, st) ← Com(sk)
3:     c := H(w, m)
4:     z := Resp(w, c, st)
5: until z ≠ ⊥
6:
7: return (w, z)
```

```
Trans(m):
1: repeat
2:     (w, st) ← Com(sk)
3:     c ← ChSet
4:     z := Resp(w, c, st)
5: until z ≠ ⊥
6: H(w, m) := c
7: return (w, z)
```

Fig. 1. Oracles Sign and Trans.

Clearly, the adversary $\mathcal{A}$ can attempt to guess w and query H on w before calling Sign/Trans, and then detect the inconsistency introduced by the reprogramming in case of Trans. However, even if the adversary makes no prior H-queries, the distribution of the random oracle changes, and this is where the gap lies. The reprogramming in Trans only reprograms the random oracle with accepting transcripts and thereby shifts the random oracle slightly towards pairs $((w, m), c)$ such that $\mathsf{Resp}(w, c, \mathsf{st}) \neq \perp$. Even though one expects this change in the distribution of the random oracle to be small, there is still a gap that needs to be properly bounded.

Both Lyubashevsky [Lyu12] and KLS [KLS18] miss the loss incurred by the bias in H in their analysis. In [Lyu12] this is missed in the hop from the real signing oracle to Hybrid 1 in the proof of Lemma 5.3 — note that the bound in [Lyu12] remains correct due to a loose analysis. In [KLS18] the gap is missed in the game hop from G_0 to G_1 in the proof of Theorem 3.2. Moreover, this oversight is not a problem limited to [Lyu12] and [KLS18], and it potentially affects all FS-based schemes involving rejection sampling. This includes a long list of works [LNP22, DKL$^+$18, DFG19, BKP20, BDK$^+$22] on lattice-based and isogeny-based signature schemes (and non-interactive proof systems) that need to be re-examined carefully.

Brigding the Gap. Our second contribution is a new, fixed proof for the CMA-to-NMA reduction for FSwA in general, and for Dilithium in particular. We address both the ROM and the QROM case; in order to optimize the reduction loss, we use slightly different (lower level) hybrids for the two cases.[2]

In order to circumvent the gap (while keeping the reduction loss reasonable), we follow a rather different (but in some sense also more natural) proof strategy than [KLS18]. We present a high-level outline of the proof (which is the same for ROM and QROM) in Section 3. The proof requires fine-grained control of the modifications to the random oracle, which we handle using nested hybrid arguments. In order to deal with QROM adversaries, we make use of the compressed-oracle technique [Zha19]. However, special care has to be taken

[2] We note that for simplicity, we consider ordinary unforgeability. It is not too hard to extend our results to *strong* unforgeability if the considered IDS satisfies the additional property of having computational unique-responses.

to deal with the potentially unbounded number of random oracle queries done by the signing procedure, as a result of the unbounded rejection sampling loop; moreover, this number depends on the choice of the message to be signed, which is under the adversary's control.

Result-wise, we note that our CMA-to-NMA reduction differs from the (flawed) one in [KLS18] in that we can rely on a weaker variant of HVZK than in [KLS18]. On the downside, the bound we obtain for the CMA-to-NMA reduction is worse than the one claimed in [KLS18]. For this reason, we conclude in Section 7 with an analysis of the security loss incurred by our proof for concrete parameters — the analysis is close to that given in [KLS18], but we improve the analysis of relevant entropy metrics — and confirm that the parameters in the Dilithium NIST submission [DKL+21] provide sufficient slack to accommodate the additional loss and still comfortably reach the claimed security for all considered NIST security levels (2, 3, and 5).

Machine-Checked Proof. We mechanize the entire security proof of Dilithium in the ROM using EasyCrypt.[3] The formalization covers the fixed CMA-to-NMA reduction (Section 5), the correctness of an HVZK simulator for the IDS underlying Dilithium, and the reduction from NMA security to MLWE and SelfTargetMSIS. The latter two proofs largely follow the original proofs in [KLS18] and are described in Section 6. These results guarantee the absence of additional gaps in the ROM proof and, due to their similarity, give high confidence that such gaps also do not exist in the QROM proof. In fact, in Section 3 we show that the two proofs have the same overall structure and that the only significant differences lie in how the probability of bad events is bounded in the ROM and the QROM.

The intricacy of the security proof, particularly the new CMA-to-NMA reduction, posed interesting challenges when formalizing the proof in EasyCrypt (even in the ROM). Indeed, the mechanized proof uses several tools that were not used in earlier mechanized cryptographic proofs.

– Proving an advantage bound that matches the pen-and-paper proof implies reasoning about the expected number of iterations of the unbounded rejection sampling loop in Dilithium. To do this, we make use of an expectation logic that was recently added to EasyCrypt to reason about the expected complexity of randomized programs [ABG+23]. The logic is based on the seminal work by Kozen [Koz83].
– Some hybrid arguments in the proof modify the operation of the rejection sampling loop one iteration at the time, which means that the total number of hybrid steps is potentially infinite. In consequence, we need to prove the convergence of advantage expressions that result from putting together all the hybrid steps, as the number of hybrid steps goes to infinity.
– In addition to various minor additions to existing EasyCrypt libraries (e.g., for limits of sequences and sums, or for conditional sampling) we developed a new matrix library supporting variable-width matrices and vectors as well as

[3] Support for QROM in EasyCrypt is still under active development [BBF+21], and the existing features do not yet allow to formally verify the QROM proof.

block matrices.[4] For the application to Dilithium, we created a new library that refines the existing EasyCrypt support for abstract polynomial rings modulo an ideal. This was necessary to express and prove low-level properties that justify some of the optimizations in Dilithium.

Altogether, the machine-checked security proof is about 6000 lines long. In addition, the generic library extensions also amount to several thousand lines. The EasyCrypt development, along with documentation on where to find the various theorem statements and how to automatically machine-check the proofs, is available at `https://github.com/formosa-crypto/dilithium`.

Concurrent Work. Concurrent and independent work [DFPS23] partially overlaps with the results we present in this paper. Both our work and [DFPS23] identify the same gap in the CMA-to-NMA reduction that is present in prior works on Fiat-Shamir with aborts. Furthermore, both [DFPS23] and our work offer new, corrected CMA-to-NMA reductions (both in the ROM and QROM), where the high-level strategy to fix the previous proofs involves reprogramming the random oracle both on accepted and rejected transcripts. But then, the two works proceed differently. [DFPS23] considers an HVZK simulator for the underlying IDS that can be used simultaneously for reprogramming accepted and rejected transcripts; such a simulator is then constructed for a particular class of signatures. On the other hand, in this paper we introduce an additional hybrid step that removes the reprogrammings of the rejected transcripts, which allows us to rely on a weaker HVZK simulator that only needs to simulate accepting transcripts. Finally, beyond the above, [DFPS23] and our work include the following respective disjoint contributions: [DFPS23] identifies and discusses some further difficulties with the Fiat-Shamir with aborts paradigm, e.g., with the history-free approach from [KLS18], and with termination and correctness in the unbounded case. On the other hand, we offer a fully mechanized security proof for Dilithium (for the classical ROM setting) using the EasyCrypt formal-verification platform.

2 Preliminaries

We consider a signature scheme obtained by applying Fiat-Shamir with aborts (FSwA) to an interactive identification scheme (IDS) that follows the standard commit-challenge-response structure. The latter means that for a public/secret key pair (pk, sk), the scheme works in three flows: 1) the Prover generates a *first message* $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$ (sometimes also called the *commitment*), and sends w to the Verifier; 2) the Verifier choses a random *challenge* $c \leftarrow C$ and sends it back to the Prover; 3) the Prover computes a *response* $z := \mathsf{Resp}(sk, w, c, \mathsf{st})$, which the Verifier checks using $\mathsf{Verify}(pk, w, c, z)$.[5] We write KeyGen for the algorithm that generates the key pair (pk, sk).

[4] This was done in collaboration with Oskar Goldhahn and has now been merged into the EasyCrypt standard library

[5] Throughout the paper, when clear from the context, we often omit the dependence on pk and sk in our notation.

The Fiat-Shamir transformation turns such an IDS into a signature scheme by computing the challenge c as the hash of w and the to-be-singed message m. We stress that by considering FSwA, we allow the IDS to abort, i.e., Resp to output $z = \perp$; in this case, the signing procedure will simply retry with a fresh new first message w until it succeeds (see Sign in Fig. 1 or 2). For a given key pair (pk, sk), we let the abort probability for w generated by Com and a random challenge c be

$$p_{(pk,sk)} := \Pr_{\substack{(w,\mathsf{st})\leftarrow\mathsf{Com}(sk)\\ c\leftarrow C}} [\mathsf{Resp}(w, c, \mathsf{st}) = \perp] \,.$$

The entropy of w will be an important parameter, implicitly captured by the guessing probability

$$\epsilon_{(pk,sk)} := \max_{w_0\in W} \Pr_{(w,\mathsf{st})\leftarrow\mathsf{Com}(sk)} [w = w_0] \,. \tag{1}$$

where W is the support set for IDS commitments. Finally, we require the IDS to satisfy the following honest-verifier zero-knowledge variant, which admits to simulate *accepted* transcripts.[6]

Definition 1. (Accepting Honest-verifier Zero-knowledge) An IDS as above is said to be acHVZK with simulation error ζ_{zk} if there exists a poly-time algorithm ZKSim that, when given the public key pk, outputs (w, c, z) with a distribution that has statistical distance at most ζ_{zk} from the distribution of a transcript (w, c, z) produced by an honest execution of the protocol *conditioned on* $z \neq \perp$.

We note that this is a different flavor of HVZK than naHVZK considered in [KLS18], and it is weaker (at least in spirit). In [KLS18] the simulator must match the full distribution of traces, which means that a (strict or expected) poly-time naHVZK simulator implies an *expected* poly-time simulator as we require it: the acHVZK simulator repeatedly runs the naHVZK simulator until a good trace is generated. Whether the acHVZK simulator is strict or expected poly-time will determine whether we require the computational hardness assumption to hold for strict or expected poly-time algorithms.[7] E.g., the scheme considered in [Lyu12] admits a strict poly-time acHVZK simulator, while for Dilithium we only know how to simulate accepted transcripts in expected poly-time.

3 Outline of the Proof

In this section, we provide a detailed account of how we closed the gap in the proof described in the introduction. We first give some intuition about the general proof strategy, and we pinpoint the main two technical steps of the proof,

[6] For simplicity, and since this is sufficient for out main application (Dilithium), we consider statistical indistinguishability of the simulated transcript. Our results extend to a computational variant in the obvious way.

[7] Also note that, at the cost of an increased simulation error, an expected poly-time simulator can always be turned into a strict poly-time one by cutting the runtime.

i.e., we isolate two quantities (corresponding to two distinguishing advantages for some game hops) that remain to be bounded. We then discuss the challenges in bounding these quantities, and we provide some intuition on how we solve them. The rigorous analyses of these quantities are then done in subsequent sections, separately for the QROM and the mechanized ROM proof.

Below, we consider an IDS as considered above, which satisfies Def. 1, and the goal is to show that EF-NMA security implies EF-CMA security for the signature scheme that is obtained from the IDS via FSwA.

3.1 Proof Skeleton

We follow the common approach, which is to show that for any CMA attacker $\mathcal{A}^{\mathsf{Sign},H}$, which has access to a signing oracle Sign and the random oracle H, one can replace the signing oracle Sign by an oracle Sim that does not have the secret key, but instead produces a valid transcript by using the acHVZK-simulator and reprograms H to be consistent with the transcript (see the description of Sim in Fig. 2 below). Turning $\mathcal{A}^{\mathsf{Sim},H}$ into an NMA attacker $\mathcal{B}^H$ that does not ask signature queries (and does not reprogram H and produces forgeries consistent with H) is then a standard argument (discussed in more detail further down).

In order to show that replacing Sign by Sim has little effect, we introduce two hybrid oracles Prog and Trans, as specified in Fig. 2, and we show that

$$\mathcal{A}^{\mathsf{Sign},H} \approx \mathcal{A}^{\mathsf{Prog},H} \approx \mathcal{A}^{\mathsf{Trans},H} \approx \mathcal{A}^{\mathsf{Sim},H} .$$

The oracle Prog samples transcripts (w, c, z) of the IDS for randomly chosen challenges c and then reprograms H consistently (denoted $H(w, m) := c \leftarrow C$), *both* for rejected *and* accepted transcripts. We emphasize that, since the reprogramming happens independently of whether the transcript is accepted or not, there is no dependency between w and c, circumventing the issue in [KLS18]. Intuitively, in order to notice the difference, $\mathcal{A}$ must have queried H on one of the points (w, m) before H gets reprogrammed on it; this is unlikely if w has high entropy.

The oracle Trans is as Prog, except that it only reprograms H on the final accepted transcript. This modification to the game introduces a bias in H towards accepting transcripts. However, this should remain unnoticed unless $\mathcal{A}$ queries such a pair (w, m) where Trans reprograms H yet Prog does not. Because w is chosen with high-entropy and not revealed to $\mathcal{A}$, this this is unlikely to happen.

Finally, closeness of $\mathcal{A}^{\mathsf{Trans},H}$ and $\mathcal{A}^{\mathsf{Sim},H}$ follows by definition of the acHVZK property: for each of the calls $\mathcal{A}$ makes to Trans, replacing it by a call to Sim changes the output distribution of $\mathcal{A}$ by at most ζ_{zk}.

The key part of the proof is bounding the loss incurred by the hops to $\mathcal{A}^{\mathsf{Prog},H}$ and $\mathcal{A}^{\mathsf{Trans},H}$, which we will do separately for the QROM proof (Section 4) and the mechanized ROM proof (Section 5). Here, we rigorously define those quantities and explain the arguments that are common to both proofs.

For any $0 \leqslant \epsilon$ and $p < 1$, for any key pair (pk, sk) with $p_{(pk,sk)} \leqslant p$ and $\epsilon_{(pk,sk)} \leqslant \epsilon$, and for any choice of $q_S, q_H \in \mathbb{N}$, let the quantities $\Delta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}(q_S, q_H)$

```
Sign(m):
1: repeat
2:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3:     $c := H(w, m)$
4:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: until $z \neq \bot$
6: return $(w, z)$
```

```
Prog(m):
1: repeat
2:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3:     $H(w, m) := c \leftarrow C$
4:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: until $z \neq \bot$
6: return $(w, z)$
```

```
Trans(m):
1: repeat
2:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3:     $c \leftarrow C$
4:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: until $z \neq \bot$
6: $H(w, m) := c$
7: return $(w, z)$
```

```
Sim(m):
1: $(w, c, z) \leftarrow \mathsf{ZKSim}(pk)$
2: $H(w, m) := c$
3: return $(w, z)$
```

Fig. 2. Overview of the different oracles used for the hybrid proof.

and $\Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}(q_S, q_H)$ be monotone in p and in ϵ, bounded from above by 1, and so that

$$\Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}(q_S, q_H) \geqslant \left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Sign},H}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prog},H}\right]\right| \qquad \text{and}$$

$$\Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}(q_S, q_H) \geqslant \left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prog},H}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Trans},H}\right]\right|$$

for any (classical or quantum) oracle algorithm $\mathcal{A}^{\mathsf{Sign},H}$ that makes at most q_S classical calls to Sign and q_H (classical or quantum) calls to the random oracle H, and outputs a single bit at the end. We take it as understood here that Sign uses the considered fixed secret key sk for the public key pk given to $\mathcal{A}$, and the same for Prog and Trans.

Having control over these two parameters, we obtain the desired CMA-to-NMA reduction via the following result. We note that the statement holds both for *classical* and *quantum* $\mathcal{A}$, where the latter can make quantum queries to H (but still only classical queries to Sign), with $\mathcal{B}$ then also being classical or quantum, respectively. In order to deal with unlikely "bad" keys that give rise to values of $p_{(pk,sk)}$ and $\epsilon_{(pk,sk)}$ close to 1, which we need to avoid, the formal statement has a precondition that bounds these two quantities (in some ways) except with small probability.

Lemma 1. *Let $\epsilon \geqslant 0$ and $p < 1$, and let $\delta := \Pr[\neg\Gamma]$ for an event Γ for which*

$$\Pr[p_{(pk,sk)} \leqslant p \;\wedge\; \epsilon_{(pk,sk)} \leqslant \epsilon \,|\, \Gamma] = 1 \tag{2}$$

where the randomness is over $(pk, sk) \leftarrow \mathsf{KeyGen}$. Let $\mathcal{A}^{\mathsf{Sign},H}$ be a CMA attacker against $\mathsf{FSwA}[\mathsf{ID}, H]$ that makes q_S queries to the signing oracle Sign and q_H queries to the random oracle H. Then, there exists an NMA attacker $\mathcal{B}^H$ against $\mathsf{FSwA}[\mathsf{ID}, H]$ so that

$$\begin{aligned}\mathsf{Adv}^{\textit{EF-CMA}}(\mathcal{A}) &\leqslant \mathsf{Adv}^{\textit{EF-NMA}}(\mathcal{B}) \\ &\quad + q_S\zeta_{zk} + \Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}(q_S, q_H + 1) + \Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}(q_S, q_H + 1) + \delta\,,\end{aligned}$$

and with running time $\mathsf{TIME}(\mathcal{B}^H) \approx \mathsf{TIME}(\mathcal{A}) + q_S\mathsf{TIME}(\mathit{ZKSim})$. *If* $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon}$ *and* $\Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p,\epsilon}$ *are concave as functions in* ϵ*, then (2) can be relaxed to*

$$\mathbb{E}[\epsilon_{(pk,sk)}|\Gamma] \leqslant \epsilon \qquad \text{and} \qquad \Pr[p_{(pk,sk)} \leqslant p \,|\, \Gamma] = 1\,.$$

Proof. Let $\mathcal{A}$ be a CMA attacker against the considered scheme. For any possible choice pk of the public key, let $\tilde{\mathcal{A}}$ be the query algorithm that given a public key pk first runs $\mathcal{A}^{\mathsf{Sign},H}(pk)$ and then verifies the correctness of the produced forgery and outputs 1 if and only if it is a valid forgery, i.e., it is a valid signature and for a message on which Sign has not been queried. Note that

$$\mathsf{Adv}^{\mathsf{EF\text{-}CMA}}(\mathcal{A}) = \mathbb{E}_{pk}\Big[\Pr\big[1 \leftarrow \tilde{\mathcal{A}}^{\mathsf{Sign},H}(pk)\big]\Big]\,.$$

Moreover, due to the verification, $\tilde{\mathcal{A}}(pk)$ makes up to $q_H + 1$ queries to H. By choice of the quantities $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}$ and $\Delta^{\mathsf{Prog}\to\mathsf{Trans}}$ and by definition of the simulation error ζ_{zk}, we have that for every choice of key pair $k = (pk, sk)$

$$\Pr\big[1 \leftarrow \tilde{\mathcal{A}}^{\mathsf{Sign},H}(pk)\big] \leqslant \Pr\big[1 \leftarrow \tilde{\mathcal{A}}^{\mathsf{Sim},H}(pk)\big] + \Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p_k,\epsilon_k} + \Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p_k,\epsilon_k} + q_s\zeta_{zk}\,,$$

where we write $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p_k,\epsilon_k}$ for $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p_k,\epsilon_k}(q_S, q_H + 1)$ to simplify the notation, and the same for $\Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p_k,\epsilon_k}$. Averaging over the choice of $k = (pk, sk)$, using that

$$\mathbb{E}\left[\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p_k,\epsilon_k}\middle|\Gamma\right] \leqslant \Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon} \qquad \text{and} \qquad \mathbb{E}\left[\Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p_k,\epsilon_k}\middle|\Gamma\right] \leqslant \Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p,\epsilon} \tag{3}$$

we obtain that

$$\mathsf{Adv}^{\mathsf{EF\text{-}CMA}}(\mathcal{A}) \leqslant \mathbb{E}_{pk}\Big[\Pr\big[1 \leftarrow \tilde{\mathcal{A}}^{\mathsf{Sim},H}(pk)\big]\Big] + \Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon} + \Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon} + q_s\zeta_{zk} + \delta\,.$$

If $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon}$ is concave as functions in ϵ, then (3) can be improved to

$$\mathbb{E}\big[\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p_k,\epsilon_k}\big|\Gamma\big] \leqslant \mathbb{E}\big[\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon_k}\big|\Gamma\big] \leqslant \Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\mathbb{E}[\epsilon_k|\Gamma]}$$

by using Jensen's inequality, and the same for $\Delta^{\mathsf{Prog}\to\mathsf{Trans}}$, showing that the relaxed requirement suffices in that case.

Recall that for a random key pair (pk, sk), a run of $\tilde{\mathcal{A}}^{\mathsf{Sim},H}(pk)$ coincides with a run of the CMA game with the original attacker $\mathcal{A}$, but where the signing queries are answered by Sim instead of Sign. This implies the existence of a NMA attacker $\mathcal{B}$ with the same success probability, using standard reasoning.

Indeed, one can consider the adversary $\mathcal{B}^H(pk)$ that given a public key pk runs $\mathcal{A}^{\mathsf{Sim},H}(pk)$ and simply outputs whatever $\mathcal{A}$ returns. However, for this, $\mathcal{B}$ has to implement the programming of H. For this, $\mathcal{B}$ can simply keep a look-up table, which stores the programmed values, and then, for every query, $\mathcal{B}$ first checks the look-up table and answers with the value in there if one exists, and otherwise it queries H and forwards the result. This is straightforward in the case of classical queries, but can also be made to work for quantum queries. For every call of $\mathcal{A}$ to H, $\mathcal{B}$ makes no more than one call to H itself, and so it makes at most q_H queries to H.

Moreover, for $\mathcal{B}$ to succeed we require that the forgery (m^*, σ^*) that $\mathcal{A}$ returns is consistent with H, i.e., the validity of it is not depending on any reprogrammed points. This is guaranteed by the $\mathsf{EF\text{-}CMA}$ notion that requires that the forgery is on a fresh message: We only program H on points (w, m) where m is a message of a signing query and therefore $m \neq m^*$. We thus get that

$$\mathsf{Adv}^{\mathsf{EF\text{-}NMA}}(\mathcal{B}) = \mathbb{E}_{pk}\Big[\Pr\big[1 \leftarrow \tilde{\mathcal{A}}^{\mathsf{Sim},H}(pk)\big]\Big]$$

which then proves the claim.

□

3.2 Challenges, and How We Solve Them

The challenges that arise in bounding $\varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}$ and $\varDelta_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}$ — and also our solutions — apply independently of whether $\mathcal{A}$ is classical or quantum. The case of a classical $\mathcal{A}$ is conceptually simpler in that we can see the random oracle as using standard lazy sampling and the elementary steps in the proof are argued using up-to-bad reasoning: we define a series of hybrids, where two consecutive games are identical until a *bad* event is triggered. This bad event typically corresponds to the adversary being able to observe a change in the distribution of a single value sampled by the random oracle. The proof then follows from proving an upper-bound on the probability of each bad event occurring and aggregating these bounds into a global advantage term.

In the case of a quantum $\mathcal{A}$, we resort to the compressed oracle technique, which can be understood as a quantum version of lazy sampling. In this setting, a *bad* event as above may then be defined via a *measurement* (we expand on this analogy in Section 4). Such a measurement typically disturbs the state, and thus the continuation of the experiment. However, thanks to the gentle-measurement lemma, if the probability of the event occurring is small (which follows from pretty much the same argument as classically) we immediately know that this disturbance is small as well. Thus, conceptually, there is no big difference in the argument for a classical and for a quantum $\mathcal{A}$. However, and interestingly, it turns out that in order to optimize the respective bounds on $\varDelta_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}$, we have to use slightly different approaches in the ROM and in the QROM proofs.

We outline the proofs of the two non trivial hops next.

The 'Program always' game hop. To bound $\varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}$, the game hop from $\mathcal{A}^{\mathsf{Sign},H}$ to $\mathcal{A}^{\mathsf{Prog},H}$ is broken down into multiple steps and substeps. At the top level, the q_S calls to Sign are replaced by calls to Prog *one by one*. For each such replacement, the challenge lies in the fact that there is no fixed upper bound on the number of loop iterations executed by the modified Sign oracle query, and thus on the number of reprogrammings that must be dealt with. Even worse, per-se, $\mathcal{A}$ could potentially affect the number of loop iterations by choosing m dependent on responses to prior H-queries. To deal with this, for each replacement of Sign by Prog, we do the replacement gradually by replacing the loop body in the query to Sign by the content of the loop body in Prog *one* *iteration at a time*. I.e., we consider the hybrid Hyb^k, which programs $H(w, m)$ to a fresh random c for the first k iterations of the loop and sets $c := H(w, m)$ for the remaining ones (see Fig. 3, middle). Thus, $\mathsf{Hyb}^0 = \mathsf{Sign}$ and $\mathsf{Hyb}^\infty = \mathsf{Prog}$.

```
Sign(m):
1: repeat
2:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3:     $c := H(w, m)$
4:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: until $z \neq \bot$
6: return $(w, z)$
```

```
Hyb^k(m):
1: $i := 0$
2: repeat
3:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
4:     if $i < k$ then
5:         $H(w, m) := c \leftarrow C$
6:     else
7:         $c := H(w, m)$
8:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
9:     $i := i + 1$
10: until $z \neq \bot$
11: return $(w, z)$
```

```
Prog(m):
1: repeat
2:     $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3:     $H(w, m) := c \leftarrow C$
4:     $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: until $z \neq \bot$
6: return $(w, z)$
```

Fig. 3. The oracles Sign (left) and Prog (right), and Hyb^k in-between (middle).

An important observation at this point is that we can exploit that the probability of remaining in the loop becomes exponentially smaller for increasing k — i.e., Hyb^k and Hyb^{k+1} become harder to distinguish — because the iteration where they could differ is less likely to be reached. In particular, the probability that round k (counting from 0) is reached, which is the round where Hyb^k and Hyb^{k+1} differ, is p^k — we stress that we crucially exploit here that in the previous rounds the challenge c was chosen at random (and not computed via H); this ensures that $\mathcal{A}$ cannot influence this probability by choosing m one or another way. Furthermore, if this round is reached then $\mathcal{A}$ can notice the difference between the two hybrids only if it has made a prior H-query to the point (w, m) where Hyb^{k+1} reprograms H while Hyb^k does not.[8] However, since w has high min-entropy, this is unlikely to have occurred. Altogether, one replacement of Sign by Prog thus incurs an error that is bounded by an infinite geometric series, for which there is the high-school closed formula. Multiplying the result with q_S, to account for the q_S times we replace Sign by Prog, we then get the desired bound on $\Delta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}$.

The quantum case is slightly trickier in that we cannot directly "inspect" prior H-queries to see if the point (w, m), where Hyb^{k+1} reprograms H while Hyb^k does not, has been queried before by $\mathcal{A}$. However, one can mimic this line of reasoning using the compressed oracle technique and doing a certain measurement, which is likely to give the desired outcome again due to the high entropy of w; furthermore, the gentle measurement lemma then ensures that the measurement introduces little disturbance. The quantitative difference to the classical case is that conditioned on reaching iteration k, the distinguishing advantage (essentially) gets a square-root, but of course the probability of reaching that

[8] Or, if H got reprogrammed on (m, w) already during a prior call to Prog.

iteration remains to be p^k, and so we end up with a similar, though slightly worse, infinite geometric series.

THE 'PROGRAM ONCE' GAME HOP. The bounding of $\mathit{\Delta}_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}$ is handled differently in the QROM and the ROM, in order to optimize the respective bounds. For the classical proof, the structure of the hybrids is the same as in the previous hop, in that we replace Prog by Trans *one by one*, and for each replacement we do it gradually *one iteration at a time*. This will then give rise to a similar infinite geometric series. The main difference to before is that if the crucial iteration (where one hybrid reprograms H at the point (m, w) and the other does not) is reached, then the reasoning for why the distinguishing advantage is small is different. Here, we rely on the fact that in order to notice the difference, $\mathcal{A}$ must make a future H-query to (m, w), but since w has high min-entropy *and* is not revealed to $\mathcal{A}$, this is unlikely to happen.

In principle, a similar strategy can be applied in the QROM setting. However, the "inspection" of future H-queries will require a measurement for every future H-query, which will lead to a unnecessarily large loss. Instead, we will do a slight detour involving a "clone" H' of H, and a variant of Prog (see Fig. 4) that also reprograms H', but only on the accepted (m, w), and then the hybrid works by replacing $\mathcal{A}$'s calls to H by calls to H' one by one.

The detailed bounds and proofs are given in Sections 4 and 5.

4 Proof in the Quantum Random Oracle Model

Here, we provide the technical details of the CMA-to-NMA reduction for the considered signature scheme obtained via Fiat-Shamir with aborts, in the QROM. As explained in Sect. 3, this boils down to bounding $\mathit{\Delta}_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}$ and $\mathit{\Delta}_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}$, and applying Lemma 1.

We start by introducing some notation and recalling a couple of elementary concepts in the context of quantum information (Sect. 4.1), and by introducing an abstract distance measure for oracles (Def. 2 in Sect. 4.2) that captures the indistinguishability of two oracles in the QROM.

4.1 Preliminaries

Let ρ_{E} be a density operator. We can speak of a (classical) event $\mathit{\Gamma}$, if ρ_{E} decomposes into $\rho_{\mathsf{E}} = \Pr[\mathit{\Gamma}]\rho_{\mathsf{E}|\mathit{\Gamma}} + \Pr[\neg\mathit{\Gamma}]\rho_{\mathsf{E}|\neg\mathit{\Gamma}}$ for probabilities $\Pr[\mathit{\Gamma}]$ and $\Pr[\neg\mathit{\Gamma}]$ that add up to 1, and density operators $\rho_{\mathsf{E}|\mathit{\Gamma}}$ and $\rho_{\mathsf{E}|\neg\mathit{\Gamma}}$. In typical cases ρ_{E} is part of a bigger state $\rho_{X\mathsf{E}}$, where X is classical, and $\mathit{\Gamma}$ is then obtained by requiring X to satisfy some property.

We will also consider events that are obtained by applying a measurement. Let ρ_{E} be a density operator and $\{P_{\mathit{\Gamma}}, P_{\neg\mathit{\Gamma}}\}$ a binary projective measurement, labeled by $\mathit{\Gamma}$ and $\neg\mathit{\Gamma}$. By default, we then write $\mathit{\Gamma}$ (and correspondingly for $\neg\mathit{\Gamma}$) for the event of observing the measurement outcome associated with P, i.e., $\Pr[\mathit{\Gamma}] = \operatorname{tr}(P_{\mathit{\Gamma}}\rho_{\mathsf{E}})$, and we let $\tilde{\rho}_{\mathsf{E}|\mathit{\Gamma}} = \frac{1}{\Pr[\mathit{\Gamma}]}P_{\mathit{\Gamma}}\rho_{\mathsf{E}}P_{\mathit{\Gamma}}$ be the corresponding post-measurement state.

We use $\delta(\rho_{\mathsf{E}}, \rho_{\mathsf{E}'}) := \frac{1}{2}\|\rho_{\mathsf{E}} - \rho_{\mathsf{E}'}\|_1$ to denote the *trace distance* between density operators ρ_{E} and $\rho_{\mathsf{E}'}$. The trace distance forms an upper bound to the advantage of any quantum algorithm in distinguishing ρ_{E} from $\rho_{\mathsf{E}'}$.

Lemma 2 (Gentle Measurement Lemma). *Let ρ_{E} be a density operator and $\{P_\Gamma, P_{\neg\Gamma}\}$ a binary projective measurement. Then $\delta(\rho_{\mathsf{E}}, \tilde{\rho}_{\mathsf{E}|\Gamma}) \leqslant \sqrt{\Pr[\neg\Gamma]}$.*

4.2 Setting Up the Stage

As explained in Sect. 3, in order to control $\Delta^{\mathsf{Sign}\to\mathsf{Prog}}_{p,\epsilon}$ and $\Delta^{\mathsf{Prog}\to\mathsf{Trans}}_{p,\epsilon}$, we consider a hybrid argument where we repeatedly replace *one* oracle call to a certain oracle by another one. To smoothen the exposition, we introduce first an abstraction of this core problem, together with a metric that captures the figure of merit.

Replacing One Oracle by Another. We consider a quantum oracle algorithm $\mathcal{A}^{H,O_1,\ldots,O_r,\mathcal{O}}$ that makes oracle calls to a random function H (i.e., a random oracle) and to arbitrary but specified oracles $O_1,\ldots,O_r$, and it makes *one* call to an *unspecified* oracle $\mathcal{O}$ (though with a specified set $\mathcal{M}$ of possible inputs), and the goal will be to show that for two particular specifications O and O', the algorithm $\mathcal{A}$ will not notice the difference whether $\mathcal{O}$ is instantiated with O or with O', i.e., that $\Pr\big[1 \leftarrow \mathcal{A}^{H,O_1,\ldots,O_r,O}\big] \approx \Pr\big[1 \leftarrow \mathcal{A}^{H,O_1,\ldots,O_r,O'}\big]$.

Considering that $\mathcal{A}$ is a quantum algorithm, we allow the queries to the random oracle H to be in superposition; for the purpose of this work, the queries to all the other oracles are classical though.

Furthermore, we note that we allow the oracle instantiations $O_1,\ldots,O_r$, as well as O and O', to also have oracle (read) access to H, and even to have oracle *write* access, i.e., they may *reprogram* H at a chosen point to a chosen value. Formally, $O_1,\ldots,O_r,O,O'$ are classical, stateless, possibly randomized oracle algorithms, with oracle *read* and *write* access to H. [9]

Closeness of Oracles In order to show indistinguishability of answering $\mathcal{A}$'s $\mathcal{O}$-query by O or O', it is sufficient to show that the output produced by $O(m)$ or $O'(m)$, together with H (which may also look different in one and the other case, due to possible different reprogramming), look alike to $\mathcal{A}$.

Using the compressed oracle technique, we can consider H to be obtained by measuring a certain quantum system D (the "compressed oracle"); namely, $H(x)$ can be obtained by measuring register D_x of D in the computational basis. Indeed the technique ensures existence of a system D, the state of which evolves (and gets entangled) upon random-oracle (superposition) queries, and that satisfies:

1. The random-oracle queries commute with measuring any of the registers of D in the computational basis. This includes reprogramming queries, which, on input (x, y) replaces the state of register D_x by $|y\rangle$.

[9] We may actually allow $O_1,\ldots,O_r$, to be stateful, all having access to the same state, but for the propose of "switching" from O to O' for any $\mathcal{A}$, this state can always be maintained and provided by $\mathcal{A}$.

2. After q (read or write) random-oracle queries, measuring all of D in the Fourier basis produces a function table that is $\hat{0}$ (sometimes denoted $\perp$) everywhere, except for up to q points. In particular, measuring all of D before any random-oracle query in the computational basis produces a uniformly random function (table) H.

The above considerations motivate to define the (parameterized) following metric, which then gives rise to the subsequent Theorem 1 .

Definition 2. *For oracle instantiations O and O' of $\mathcal{O}$, and for $q \in \mathbb{N}$,*

$$d_q(O, O') := \max_{m, \rho_{\mathsf{DE}}} \delta\big(\rho_{O(m)H\mathsf{E}}, \rho_{O'(m)H\mathsf{E}}\big)$$

where the maximum is over all possible $m \in \mathcal{M}$ and over all states ρ_{DE} with the property that D behaves as in 2. above (for the considered q) upon measuring in the Fourier basis; furthermore, $\rho_{O(m)H\mathsf{E}}$ is obtained from m and ρ_{DE} by running O on input m, and by measuring all of D in the computational basis to obtain H, and the same for $\rho_{O'(m)H\mathsf{E}}$.

It is not too hard to argue that the maximum is indeed attained in the definition of d_q. Furthermore, $q' \geqslant q \Rightarrow d_{q'} \geqslant d_q$, and d_q satisfies the triangle inequality.

To help to understand the intuition for this metric, we point out that in the corresponding classical counter part, we would maximize over all query inputs m and over all possible lazy-sampled databases D that have at most q entries, and then compare the respective distributions of $(O(m), H)$ and $(O'(m), H)$, obtained by running O, respectively O', on m, and obtaining H by filling in all the empty places in (the possibly reprogrammed) database D by random values.

The following is straightforward to prove.

Theorem 1. *Consider a quantum oracle algorithm $\mathcal{A}^{H,O_1,\ldots,O_r,\mathcal{O}}$ for arbitrary but fixed oracle instantiations $O_1, \ldots, O_r$, and let O and O' be two possible instantiations for $\mathcal{O}$, as specified above. Recall that $\mathcal{A}$ is restricted to making* one *query to $\mathcal{O}$. Let Q be the number of oracle calls to H, made by $\mathcal{A}$ and $O_1, \ldots, O_r$,* prior *to $\mathcal{A}$'s oracle call to $\mathcal{O}$.*[10] *Then,*

$$\Big|\Pr\big[1 \leftarrow \mathcal{A}^{H,O_1,\ldots,O_r,O}\big] - \Pr\big[1 \leftarrow \mathcal{A}^{H,O_1,\ldots,O_r,O'}\big]\Big| \leqslant \mathbb{E}_Q[d_Q(O, O')] \,.$$

As a toy example application, consider a quantum oracle algorithm $\mathcal{A}^{H,\mathcal{O}}$ that makes at most q_H queries to a random oracle H and $q_{\mathcal{O}}$ queries to a non-instantiated oracle $\mathcal{O}$. Let us assume that O and O' are instantiations of $\mathcal{O}$ that make no queries to H, and it holds that $d_q(O, O') \leqslant q_H \varepsilon$ for any q. Then, by repeated application of Theorem 1 in order to switch from O to O' one by one, we immediately obtain that $\big|\Pr\big[1 \leftarrow \mathcal{A}^{H,O}\big] - \Pr\big[1 \leftarrow \mathcal{A}^{H,O'}\big]\big| \leqslant q_H q_{\mathcal{O}} \varepsilon$.

[10] This includes calls to reprogram H.

Typical Strategies There are two generic approaches to prove that $d_q(O, O')$ is small:

Strategy 1. Show the existence of a (classical) event Γ (Γ for "good" event) with the property that, for any m and ρ_{DE}, (1) the event Γ has the same probability $\Pr[\Gamma]$ of occurrence when running O and O', and (2) O and O' act identically conditioned on Γ, and thus the two states $\rho_{O(m)H\mathsf{E}|\Gamma}$ and $\rho_{O'(m)H\mathsf{E}|\Gamma}$ are identical. Indeed, in that case, by basic properties of the trace distance,

$$\begin{aligned}\delta\big(\rho_{O(m)H\mathsf{E}}, \rho_{O'(m)H\mathsf{E}}\big) &\leqslant \Pr[\Gamma]\delta(\rho_{O(m)H\mathsf{E}|\Gamma}, \rho_{O'(m)H\mathsf{E}|\Gamma}) &(4)\\ &\quad + \Pr[\neg\Gamma]\delta(\rho_{O(m)H\mathsf{E}|\neg\Gamma}, \rho_{O'(m)H\mathsf{E}|\neg\Gamma}) &(5)\\ &\leqslant \Pr[\neg\Gamma]\,. &(6)\end{aligned}$$

Strategy 2. Show the existence of a binary projective measurement $\{P_\Gamma, P_{\neg\Gamma}\}$ on D and the internal state of the respective oracles, so that when applied during the run of the oracle, similarly to above, (i) the event Γ (of observing the measurement outcome associated with P_Γ) has the same probability $\Pr[\Gamma]$ of occurrence when running O and O', and (ii) O and O' act identically conditioned on Γ, and thus the two states $\tilde\rho_{O(m)H\mathsf{E}|\Gamma}$ and $\tilde\rho_{O'(m)H\mathsf{E}|\Gamma}$ are identical. Indeed, in that case, by triangle inequality,

$$\begin{aligned}\delta\big(\rho_{O(m)H\mathsf{E}}, \rho_{O'(m)H\mathsf{E}}\big) &\leqslant \delta(\rho_{O(m)H\mathsf{E}}, \tilde\rho_{O(m)H\mathsf{E}|\Gamma}) &(7)\\ &\quad + \delta(\tilde\rho_{O(m)H\mathsf{E}|\Gamma}, \tilde\rho_{O'(m)H\mathsf{E}|\Gamma}) &(8)\\ &\quad + \delta(\tilde\rho_{O'(m)H\mathsf{E}|\Gamma}, \rho_{O(m)H\mathsf{E}}) &(9)\\ &\leqslant 2\sqrt{\Pr[\neg\Gamma]} &(10)\end{aligned}$$

where the final inequality is by applying the gentle measurement lemma twice.

This extends in the obvious way to a *classically-controlled* measurement, i.e., to a measurement that is only applied if a particular classical bit b is set, and such that (o) the classical bit b is set with the same probability when running O and O', (i) conditioned on b being set, the event Γ has the same probability $\Pr[\Gamma]$ of occurrence when running O and O', and (ii) O and O' act identically conditioned on b not being set, or conditioned on b set and Γ. The above bound then becomes

$$\delta\big(\rho_{O(m)H\mathsf{E}}, \rho_{O'(m)H\mathsf{E}}\big) \leqslant 2\Pr[b=1]\sqrt{\Pr[\neg\Gamma|b=1]}\,. \tag{11}$$

4.3 Core of the Proof

We need to bound the quantities $\Delta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}(q_S, q_H)$ and $\Delta_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}(q_S, q_H)$. For that purpose, we consider a fixed key (pk, sk) for which $p_{(pk,sk)} \leqslant p$ and $\epsilon_{(pk,sk)} \leqslant \epsilon$, and we consider a quantum oracle algorithm $\mathcal{A}$ with a binary output, and which makes q_S classical queries to Sign and q_H quantum queries to the random oracle H. Our goal then is to bound the respective closeness of $\mathcal{A}^{\mathsf{Sign},H}$ and $\mathcal{A}^{\mathsf{Prog},H}$ and of $\mathcal{A}^{\mathsf{Prog},H}$ and $\mathcal{A}^{\mathsf{Trans},H}$; we do this below.

Closeness of $\mathcal{A}^{\mathsf{Sign},H}$ and $\mathcal{A}^{\mathsf{Prog},H}$ For the purpose of showing closeness of Sign and Prog, we introduce the following hybrid oracles. For every $k \in \mathbb{N}$, the oracle Hyb^k replaces the first k evaluations $c := H(w, m)$ in the loop of Sign to freshly reprogramming $H(w, m) := c \leftarrow C$, with the convention that $\mathsf{Hyb}^0 := \mathsf{Sign}$. In other words, Hyb^k acts like Prog for the first k iterations of the loop, and then like Sign for the remaining ones (if it is still looping then).

Lemma 3. *$d_q(\mathsf{Hyb}^{k-1}, \mathsf{Hyb}^k) \leqslant 2p^{k-1}\sqrt{(q+k)\epsilon}$ for every $k \geqslant 1$.*

The claim here is closely related to the *adaptive reprogramming* in [GHHM21]; however, there are some subtle technical differences (with the crucial reprogramming step being reached only with a certain probability, and with prior reprogrammings taking place). For this reason, and for consistency with the other parts of the proof, we prove Lemma 3 from scratch.

Proof. The oracle Hyb^{k-1} and Hyb^k only differ at the kth iteration, in which the former performs an evaluation $c := H(w, m)$, while the latter performs a fresh reprogramming $H(w, m) := c \leftarrow C$. We call this the crucial iteration.

We follow Strategy 2. and consider a binary projective measurement performed right before c is determined in the crucial iteration, classically controlled by the bit b that is set if the crucial iteration is executed (i.e., the loop has not stopped before). The measurement checks whether or not the sampled w in the crucial iteration is such that measuring $\mathsf{D}_{(w,m)}$ in the Fourier basis produces $\hat{0}$, and Γ is satisfied if this is the case (i.e., if (w, m) is not recorded in the database). Recall that in case the loop terminates prior to the crucial iteration, no measurement is performed.

Clearly, (o) b is set with the same probability when running Hyb^{k-1} and Hyb^k, (i) if the crucial iteration is reached then $\Pr[\Gamma]$ is the same when running Hyb^{k-1} and Hyb^k, and (ii) if the crucial iteration is reached and Γ is satisfied then, in both cases, c is uniformly random and $H(w, m)$ becomes c — in case of Hyb^k by construction, and in case of Hyb^{k-1} since c is then obtained by measuring the state $|\hat{0}\rangle$ of $\mathsf{D}_{(a,m)}$ in the computational basis — and thus Hyb^{k-1} and Hyb^k act identially. Hyb^{k-1} and Hyb^k obviously also behave the same if the crucial iteration is not reached. Thus, by (11), $d_q(\mathsf{Hyb}^{k-1}, \mathsf{Hyb}^k) \leqslant 2\Pr[b = 1]\sqrt{\Pr[\neg\Gamma | b = 1]}$. It remains to control this latter term.

For $b = 1$ to happen, the loop must have entered the kth iteration, which happens with probability $\Pr[b = 1] = p^{k-1}$ because every previous transcript is freshly sampled.

Conditioned on $b = 1$ where the loop enters the kth iteration, the database records no more than $q + k$ non-$\hat{0}$ entries, and a is freshly sampled, and thus by (1) and union bound, we obtain $\Pr[\neg\Gamma | b = 1] \leqslant (q + k)\epsilon$. This concludes the proof. □

Now applying Lemma 3 k times, together with the triangle inequality for d_q we obtain

$$d_q(\mathsf{Sign}, \mathsf{Hyb}^k) \leqslant \sum_{1\leqslant i\leqslant k} 2p^{i-1}\sqrt{(q+i)\epsilon} \leqslant \frac{2\sqrt{\epsilon}}{1-p}(1-p)\sum_{i\geqslant 1} p^{i-1}\sqrt{(q+i)}$$
$$\leqslant \frac{2\sqrt{\epsilon}}{1-p}\sqrt{(1-p)\sum_{i\geqslant 1} p^{i-1}q + (1-p)\sum_{i\geqslant 1} p^{i-1}i} \leqslant \frac{2\sqrt{\epsilon}}{1-p}\sqrt{q+\frac{1}{(1-p)}}\,, \quad (12)$$

where the third inequality is Jensen's inequality (exploiting that, by the standard formula for a geometric series, $(1-p)\sum_i p^{i-1} = 1$), and the last one follows by again applying the standard formula for a geometric series (noting that the second term is the derivative of a geometric series).

Next, we argue closeness of Hyb^k to Prog. Recall that Prog is obtained from Sign by replacing every evaluation $c := H(w, m)$ in the loop to a fresh reprogramming $H(w, m) := c \leftarrow C$, whereas Hyb^k does so only for the first k iterations.

Lemma 4. *$d_q(\mathsf{Hyb}^k, \mathsf{Prog}) \leqslant p^k$ for every $k \in \mathbb{N}$.*

Proof. We follow Stragtegy 1 and define the good event Γ where a non-abort response $z \neq \perp$ is output within the first k iterations in a call to $\mathsf{Hyb}^k/\mathsf{Prog}$. Indeed, (i) the probability $\Pr[\neg\Gamma]$ depends only on the first k iterations, hence it is the same in both oracles, and (ii) conditioned on Γ, the loop terminates within k iterations, so that both oracles behave identically. Since within each iteration the transcript (w, c, z) is freshly sampled, the probability that all k iterations yield $z = \perp$ is bounded by $\Pr[\neg\Gamma] \leqslant p^k$. Thus, by (4) we conclude that $d_q(\mathsf{Hyb}^k, \mathsf{Prog}) \leqslant \Pr[\neg\Gamma] \leqslant p^k$. □

By combining (12) and Lemma 4, and letting k go to infinity, we obtain

$$d_q(\mathsf{Sign}, \mathsf{Prog}) \leqslant \frac{2\sqrt{\epsilon}}{1-p}\sqrt{q+\frac{1}{1-p}}\,.$$

Replacing every invocation of Sign in $\mathcal{A}^{\mathsf{Sign},H}$ by Prog from left to right, and further taking into account that the expected number of read/write queries to H prior to every replacement is $\mathbb{E}[Q] \leqslant q_H + (q_S - 1)/(1-p)$, we apply Theorem 1 q_S times and obtain

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Sign},H}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prog},H}\right]\right| \leqslant q_S \cdot \mathbb{E}_Q[d_Q(\mathsf{Sign}, \mathsf{Prog})]$$
$$\leqslant q_S \cdot \mathbb{E}_Q\left[\tfrac{2\sqrt{\epsilon}}{1-p}\sqrt{Q+\tfrac{1}{1-p}}\right] \leqslant \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{\mathbb{E}_Q[Q]+\frac{1}{1-p}}$$
$$\leqslant \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{q_H+\frac{q_S}{1-p}}\,,$$

where the third inequality is by Jensen's inequality. This proves Corollary 1.

Corollary 1. $\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Sign},H}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prog},H}\right]\right| \leqslant \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{q_H + \frac{q_S}{1-p}}\,.$

Closeness of $\mathcal{A}^{\mathsf{Prog},H}$ and $\mathcal{A}^{\mathsf{Trans},H}$. For the purpose of showing closeness of $\mathcal{A}^{\mathsf{Prog},H}$ and $\mathcal{A}^{\mathsf{Trans},H}$, we introduce a second instantiation H' of the random oracle, which is set to be equal to H at the beginning, and we modify Prog to Prog' so as to also reprogram H', but only on the accepted transcript (see Fig. 4 middle). Looking ahead, we notice that this detour via Prog' and H' is not done in the ROM proof; there, we have a (more) direct argument to go from $\mathcal{A}^{\mathsf{Prog},H}$ to $\mathcal{A}^{\mathsf{Trans},H}$, very similar to the one going from $\mathcal{A}^{\mathsf{Sign},H}$ to $\mathcal{A}^{\mathsf{Prog},H}$. The reason we do it this way here is that we obtain a better bound than when trying to mimic the reasoning that is used in the ROM proof.

```
Prog(m):
1: repeat
2:     (w, st) ← Com(sk)
3:     H(w, m) := c ← C
4:     z := Resp(w, c, st)
5: until z ≠ ⊥
6: return (w, z)
```

```
Prog'(m):
1: repeat
2:     (w, st) ← Com(sk)
3:     H(w, m) := c ← C
4:     z := Resp(w, c, st)
5: until z ≠ ⊥
6: H'(w, m) := c
7: return (w, z)
```

```
Trans(m):
1: repeat
2:     (w, st) ← Com(sk)
3:     c ← C
4:     z := Resp(w, c, st)
5: until z ≠ ⊥
6: H(w, m) := c
7: return (w, z)
```

Fig. 4. The oracles Prog, Prog' and Trans.

Since the adverary $\mathcal{A}$ in an execution of $\mathcal{A}^{\mathsf{Prog},H}$ has its random-oracle queries answered by H, and $\mathcal{A}$ has no access to H', we obviously have that $\mathcal{A}^{\mathsf{Prog},H} = \mathcal{A}^{\mathsf{Prog}',H}$. Similarly, $\mathcal{A}^{\mathsf{Prog}',H'} = \mathcal{A}^{\mathsf{Trans},H}$. Thus, it remains to show closeness of $\mathcal{A}^{\mathsf{Prog}',H}$ and $\mathcal{A}^{\mathsf{Prog}',H'}$. Towards this goal, we first settle the following properties of an execution of Prog'.

Proposition 1. *For an arbitrary but fixed message m_0, let (w, c, z) be the first non-$\perp$ transcript produced in an invocation of $\mathsf{Prog}'(m_0)$, and let S' be the set of w's sampled in the loop for which $z = \perp$. Then the following holds.*

– *The distribution of (S', w, c, z) is invariant to the choice of m_0.* (13)
– *S' is statistically independent of (w, c, z).* (14)
– *For every $w^0 \in A$, $\Pr[w^0 \in S'] \leqslant \frac{\epsilon}{1-p}$.* (15)

Proof. Let $t_i = (w_i, c_i, z_i)$ be the transcript sampled in the i-th iteration of the loop. For the purpose of the analysis, we assume that t_i is sampled for *every* $i \in \mathbb{Z}_{>0}$, even if the loop stops before. Then, the t_i's are i.i.d. distributed, and S' equals $\{w_1, \ldots, w_{K-1}\}$, with K being minimal such that $z_K \neq \perp$ and $(w, c, z) = t_K$. As the sampling of (S', w, c, z) does not involve m_0 at all, (13) follows immediately.

For the analysis of (14), we consider the list $L := [t_1, \ldots, t_{K-1}]$; clearly showing independence of L and (w, c, z) implies independence of S' and (w, c, z). Further consider an arbitrary but fixed list $L^0 = [t^0_1, \ldots, t^0_{k-1}]$ of transcripts

$t_i^0 = (w_i^0, c_i^0, z_i^0)$, and an arbitrary but fixed transcript $t^0 = (w^0, c^0, z^0)$. With the goal to show that

$$\Pr\left[L = L^0 \text{ and } (w,c,z) = t^0\right] = \Pr\big[L = L^0\big] \cdot \Pr\big[(w,c,z) = t^0\big] \,, \tag{16}$$

we may assume $z_1^0 = \cdots = z_{k-1}^0 = \bot$ and $z^0 \neq \bot$, because otherwise both sides of (16) vanish trivially. But then, by definition of L and (w,c,z),

$$\begin{aligned}
&\Pr\left[L = L^0 \text{ and } (w,c,z) = t^0\right] = \Pr\left[\forall i < k : t_i = t_i^0 \text{ and } t_k = t^0\right] \\
&= \Pr\left[\forall i < k : t_i = t_i^0 \text{ and } t_k = t^0 \text{ and } z_k \neq \bot\right] \\
&= \Pr\begin{bmatrix} z_k \neq \bot \\ \forall i < k : t_i = t_i^0 \end{bmatrix} \cdot \Pr\left[t_k = t^0 \,\middle|\, \begin{matrix} z_k \neq \bot \\ \forall i < k : t_i = t_i^0 \end{matrix}\right] \\
&= \Pr\begin{bmatrix} z_k \neq \bot \\ \forall i < k : t_i = t_i^0 \end{bmatrix} \cdot \Pr\left[t_k = t^0 \middle| z_k \neq \bot\right] \\
&= \Pr\left[L = L^0\right] \cdot \Pr\left[t_k = t^0 \middle| z_k \neq \bot\right] \,,
\end{aligned}$$

where the fourth equality is due to independence between $(t_1, \ldots, t_{k-1})$ and t_k. Furthermore, summing up both sides of the above equality over all choices of L^0, noting that $\Pr\left[t_k = t^0 \middle| z_k \neq \bot\right]$ does not depend on k (since the t_i's are i.i.d.), we immediately get that $\Pr\left[t_k = t^0 \middle| z_k \neq \bot\right] = \Pr\big[(w,c,z) = t^0\big]$, which shows (16) and thus (14).

Next, notice that $|L| \geqslant \ell$ implies $z_1 = \cdots = z_\ell = \bot$. Thus,

$$\begin{aligned}
&\Pr\big[a^0 \in S'\big] \leqslant \sum_{\ell \geqslant 1} \Pr\big[w_\ell = w^0 \text{ and } |L| \geqslant \ell\big] \\
&\leqslant \sum_{\ell \geqslant 1} \Pr\big[w_\ell = w^0 \text{ and } z_1 = \cdots = z_{\ell-1} = \bot\big] \\
&= \sum_{\ell \geqslant 1} \Pr\big[w_\ell = w^0\big] \cdot \Pr\big[z_1 = \cdots = z_{\ell-1} = \bot\big] \leqslant \sum_{\ell \geqslant 1} p^{\ell-1}\epsilon \leqslant \frac{\epsilon}{1-p} \,,
\end{aligned}$$

where the equality holds due to the independence between a_ℓ and $(z_1, \ldots, z_{\ell-1})$. This concludes (15). ◻

For this purpose, for every $0 \leqslant i \leqslant q_H$ we let $\mathcal{G}_i$ be the hybrid between $\mathcal{A}^{\mathsf{Prog}',H}$ and $\mathcal{A}^{\mathsf{Prog}',H'}$ that has the first i queries to the random oracle answered by H', and the remaining ones by H. Obviously, $\mathcal{G}_0 = \mathcal{A}^{\mathsf{Prog}',H}$, while $\mathcal{G}_{q_H} = \mathcal{A}^{\mathsf{Prog}',H'}$. Thus, considering an arbitrary but fixed $1 \leqslant i \leqslant q_H$ and setting $\mathcal{G} := \mathcal{G}_{i-1}$ and $\mathcal{G}' := \mathcal{G}_i$, it is sufficient to show that $\mathcal{G}$ and $\mathcal{G}'$ are close. This is indeed the case:

Lemma 5. $\big|\Pr[1 \leftarrow \mathcal{G}] - \Pr[1 \leftarrow \mathcal{G}']\big| \leqslant 2\sqrt{\frac{q_S \epsilon}{1-p}}$.

Proof. Below, we refer to the i-th query of $\mathcal{A}$ to the random oracle, i.e., the query on which $\mathcal{G}$ and $\mathcal{G}'$ differ, as the *crucial query.*

In the respective executions of $\mathcal{G}$ and $\mathcal{G}'$, we define S as the set of all the w's that Prog' sampled but for which $\mathsf{Resp}(w, c, \mathsf{st}) = \bot$, in all the invocations of Prog' before the crucial query. Thus, by construction, at the time of the crucial query, H and H' differ at most at the points in S. (They might agree on a point in S, if the freshly sampled value for H at this point equals the old value.)

For the sake of analysis, consider a binary projective measurement on the input query register for the crucial query, which measures whether or not the input (w, m) is such that $w \in S$. Let Γ be satisfied if $w \notin S$, and let $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$ be the two respective games obtained by performing this measurement. Since H and H' only differ at the places (w, m) where $w \in S$, conditioned on Γ, the two oracles behave identically, and thus do $\tilde{\mathcal{G}}$ and $\tilde{\mathcal{G}}'$. Furthermore, the probability $\Pr[\Gamma]$ is the same in both games.

Thus by a double application of the gentle measurement lemma, we have

$$\begin{aligned}|\Pr[1 \leftarrow \mathcal{G}] - \Pr[1 \leftarrow \mathcal{G}']| \leqslant{} & |\Pr[1 \leftarrow \mathcal{G}] - \Pr[1 \leftarrow \tilde{\mathcal{G}}' \mid \Gamma]| \\ & + |\Pr[1 \leftarrow \tilde{\mathcal{G}}' | \Gamma] - \Pr[1 \leftarrow \tilde{\mathcal{G}} | \mid \Gamma]| \\ & + |\Pr[1 \leftarrow \tilde{\mathcal{G}} | \Gamma] - \Pr[1 \leftarrow \mathcal{G}]| \\ \leqslant{} & 2\sqrt{\Pr[\neg\Gamma]}\,.\end{aligned}$$

Hence, it remains to bound the probability $\Pr[\neg\Gamma]$. The intuition is that S collects those w's that Prog dismisses; thus, $\mathcal{A}$ does not get to see them, so it is hard for him to find an element in S, hence Γ is satisfied most likely. However, turning this intuition into a rigorous argument is not fully straightforward, since the set S, as a random variable, has a somewhat odd distribution.

Let Q be the random variable indicating the number of queries made to Prog prior to the crucial query; we have with certainty that $Q \leqslant q_S$.

A crucial observation that holds for both $\mathcal{G}, \mathcal{G}'$ is that, conditioned on $Q = q$ for an arbitrary but fixed q, the set S equals $S'_1 \cup \cdots \cup S'_q$ where every S'_j is the set S' that was produced in the jth query of Prog' as specified in Proposition 1. We note that, at the time the adversary $\mathcal{A}$ makes the crucial query, H has not been queried, and (w, c, z) in Proposition 1 is the only information that is dissipated to the adversary for every prior query to Prog'. It follows from (13) and (14) that every S'_j is independent from the view of adversary, and hence so is S.

Due to the independence, it suffices to bound $\Pr[w^0 \in S | Q = q]$ for every $w^0 \in W$. Then it follows from the union bound and (15) that

$$\Pr[w^0 \in S | Q = q] \leqslant \sum_{j \in [q]} \Pr[w^0 \in S'_j] \leqslant \frac{q_S \epsilon}{1 - p}\,.$$

Putting things together, the proof is concluded. □

Corollary 2. $\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Prog}, H}\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Trans}, H}\right]\right| \leqslant 2 q_H \sqrt{\frac{q_S \epsilon}{1-p}}\,.$

4.4 Wrapping Up

From the above it follows that

$$\varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}(q_S,q_H) \leqslant \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{q_H+\frac{q_S}{1-p}} \quad \text{and} \quad \varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}(q_S,q_H) \leqslant 2q_H\sqrt{\frac{q_S\epsilon}{1-p}}$$

and thus by Lemma 1, we obtain the following.

Theorem 2. *Let $\epsilon, p, \delta < 1$ be so that there exists an event $\varGamma$ with $\Pr[\neg\varGamma] \leqslant \delta$, $\Pr[p_{(pk,sk)} \leqslant p \mid \varGamma] = 1$ and $\mathbb{E}[\epsilon_{(pk,sk)} \mid \varGamma] \leqslant \epsilon$ for $(pk, sk) \leftarrow$ KeyGen. Let $\mathcal{A}^{\mathsf{Sign},H}$ be a quantum CMA attacker against $\mathsf{FSwA}[\mathsf{ID}, H]$ that makes q_S queries to the signing oracle Sign and q_H quantum queries to the random oracle H. Then, there exists a quantum NMA attacker $\mathcal{B}^H$ so that*

$$\begin{aligned}\mathsf{Adv}^{\textit{EF-CMA}}(\mathcal{A}) \leqslant\ & \mathsf{Adv}^{\textit{EF-NMA}}(\mathcal{B}) \\ & + \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{q_H+1+\frac{q_S}{1-p}} + 2(q_H+1)\sqrt{\frac{q_S\epsilon}{1-p}} + q_S\zeta_{zk} + \delta\end{aligned}$$

and with running time $\mathsf{TIME}(\mathcal{B}^H) \approx \mathsf{TIME}(\mathcal{A}) + q_S\mathsf{TIME}(\mathit{ZKSim})$.

5 The Mechanized ROM Proof

We now describe the mechanized proof of the CMA-to-NMA reduction in the ROM. As argued in Section 3, the high-level structure is the same as in the QROM proof. We again want to instantiate Lemma 1. We assume query bounds q_S for signature queries and q_H for random oracle queries. In order to obtain the bound for the CMA-to-NMA reduction, we need to provide $\varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}}$ and $\varDelta_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}}$ and prove these bounds for an arbitrary (but fixed) key pair (pk, sk) such that $p_{(pk,sk)} \leqslant p$ and $\epsilon_{(pk,sk)} \leqslant \epsilon$. We set:

$$\begin{aligned}\varDelta_{p,\epsilon}^{\mathsf{Sign}\to\mathsf{Prog}} &:= q_S\epsilon\left(\frac{q_S+1}{2(1-p)^2}+\frac{q_H}{1-p}\right) \\ \varDelta_{p,\epsilon}^{\mathsf{Prog}\to\mathsf{Trans}} &:= \frac{q_Sq_H\epsilon}{1-p}\end{aligned}$$

Applying some simplifications, this allows us to prove the following bound.

Theorem 3. *Let $\epsilon, p, \delta < 1$ be as in Theorem 2. Let $\mathcal{A}^{\mathsf{Sign},H}$ be a classical CMA attacker against $\mathsf{FSwA}[\mathsf{ID}, H]$ that makes q_S queries to the signing oracle Sign and q_H queries to the random oracle H. Then, there exists a classical NMA attacker $\mathcal{B}^H$ against $\mathsf{FSwA}[\mathsf{ID}, H]$ so that*

$$\mathsf{Adv}^{\textit{EF-CMA}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\textit{EF-NMA}}(\mathcal{B}) + \frac{2q_S(q_H+1)\epsilon}{(1-p)} + \frac{q_S\epsilon(q_S+1)}{2(1-p)^2} + q_S\zeta_{zk} + \delta$$

and with running time $\mathsf{TIME}(\mathcal{B}^H) \approx \mathsf{TIME}(\mathcal{A}) + q_S\mathsf{TIME}(\mathit{ZKSim})$.

Proof. It suffices to show that the bounds for $\Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}$ and $\Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}$ are indeed correct; the theorem then follows with Lemma 1. For $\Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}$, as outlined in Section 3.2, we successively replace the individual loop iterations of the Sign oracle with iterations from the Prog oracle (cf. Hyb in Fig. 3). That is, we have q_S sequences of hybrid arguments (one for each query), each replacing one-by-one κ loop iterations (cf. Hyb in Fig. 3). After κ steps, we cut off the remaining loop for a loss of p^κ. This yields an intermediate game where $\mathcal{A}$ interacts with H and Prog^κ, the latter behaving like Prog but aborting after κ iterations. The final bound is then obtained as the limit when κ is increased to infinity (causing Prog^κ to become Prog). Consider the hybrid step where the queries 0 to $i-1$ are answered by Prog^κ, query i is answered by Hyb^j and all remaining queries are answered by Sign. We bound the loss of answering query i with Hyb^{j+1} instead. Assuming a lazy implementation of the random oracle, both games behave the same unless iteration j on query i is reached *and* the pair (w, m) is already in the (previously queried) domain of H. The probability of this "bad" event occurring can be bounded by

$$\delta_{i,j} := p^j \epsilon \left(\frac{i}{1-p} + q_H + j \right)$$

where the term in parentheses is an upper bound on the *expected* size of the (previously queried) domain of H at the point where the bad event might occur (i.e., iteration j of query i). In there, the term $\frac{i}{1-p}$ is the expected number of iterations of the i preceding calls. Summing the total loss over i and j we have

$$\sum_{i=0}^{q_S-1} \left(p^\kappa + \sum_{j=0}^{\kappa-1} \delta_{i,j} \right) \leqslant q_S \cdot p^\kappa + \Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}$$

which converges to $\Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}$ as κ is increased to infinity. For $\Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}$ the structure of the hybrid argument is exactly the same, the difference lies in how the bad event is bounded. Let Hyb_2 be the analog to Hyb, replacing iterations of Prog with those of Trans, and consider the replacement of Hyb_2^j with Hyb_2^{j+1} on query i. The two games behave the same, unless (a) iteration j of query i is reached and unsuccessful and (b) the adversary queries H using the pair (w, m) at some (later) point in the game. The probability of (a) is bounded by p^j and the probability of b is at most $q_H\epsilon$. Summing and taking the limit as above yields $\Delta_{p,\epsilon}^{\mathsf{Prog}\rightarrow\mathsf{Trans}}$. □

Remark 1. The theorem we formalized in EasyCrypt is slightly less general than Theorem 3. We only consider the case of a perfect simulator (i.e, $\zeta_{zk} = 0$), and we restrict to the case where the simulator is obtained by wrapping a simulator for a single run of the IDS in a while loop. These simplifications naturally match our application to Dilithium.

Mechanizing the proof of the aforementioned variant of Theorem 3 turned out to be challenging for a number of reasons. In the following, we briefly comment on the most important ones.

First and foremost, the analysis of the bad event used to establish the bound $\Delta_{p,\epsilon}^{\mathsf{Sign}\rightarrow\mathsf{Prog}}$ crucially relies on the ability to take into account the expected size of the domain of the random oracle at the point where the bad event can (potentially) occur. Even with the intermediate oracle Prog^{κ}, a worst-case assumption on the i preceding queries would give a term of $i \cdot \kappa$ instead of $\frac{i}{1-p}$, causing the sum to no longer converge as κ is increased to infinity. The expected-size analysis for the domain of the random oracle H is carried out using an expectation logic. This expectation logic is an adaptation of the seminal work by Kozen [Koz83] and was recently added to EasyCrypt to reason about the expected complexity of randomized programs. [ABG+23] Our work provides the first application of this logic to cryptographic proofs.

Moreover, while the argument for $\varDelta_{p,\epsilon}^{\mathsf{Prog} \to \mathsf{Trans}}$ is intuitively much simpler than the argument for $\varDelta_{p,\epsilon}^{\mathsf{Sign} \to \mathsf{Prog}}$, the proof in EasyCrypt is almost as complex. Unlike for $\varDelta_{p,\epsilon}^{\mathsf{Sign} \to \mathsf{Prog}}$, the bad event is not necessarily triggered during the critical iteration; it can be triggered whenever H is queried. In order to bound the probability of the bad event occurring, we exploit that—assuming that iteration j of query i is unsuccessful—the commitment w is never used. This allows us to bound the bad event by transforming the game into one where w is sampled *after* the adversary is finished. This is called an eager/lazy argument in EasyCrypt.

Lastly, the hybrid arguments for bounding $\varDelta_{p,\epsilon}^{\mathsf{Sign} \to \mathsf{Prog}}$ and $\varDelta_{p,\epsilon}^{\mathsf{Prog} \to \mathsf{Trans}}$ involve a complex interplay of up-to-bad reasoning, hybrid steps, and a limit construction that ultimately lets the number of hybrid steps approach infinity. To the best of our knowledge, such a construction has not been formalized in EasyCrypt before.

6 A Machine-Checked Security Proof for Dilithium

We now describe the machine-checked security proof for Dilithium. More precisely, we prove EF-CMA security of the "template scheme" from the specification document [DKL+21, Figure 1] extended with public key compression. This is equivalent to Dilithium-QROM [KLS18, Figure 17] with $\mathbf{A}$ and $\mathbf{y}$ sampled randomly (i.e., not generated from a seed) and with an unbounded loop for the signing procedure.

The overall structure of the machine-checked proof largely follows [KLS18]. We first prove EF-NMA security by a reduction from MLWE and SelfTargetMSIS. We then express Dilithium as the FSwA transform of an IDS and provide an HVZK simulator for this IDS. This allows us to instantiate Theorem 3 and conclude EF-CMA security of Dilithium.

6.1 Dilithium Specification

Most of the operations in Dilithium operate on vectors and matrices over the rings $R := \mathbb{Z}[X]/(X^n+1)$ and $R_q := \mathbb{F}_q[X]/(X^n+1)$. The specification [DKL+21] sets n to 256 and q to the prime $8380417 = 2^{23} - 2^{13} + 1$. In addition, there are a number of supporting algorithms (e.g., highBits or makeHint) that deal with certain kinds of rounding.

While the specification is written for one (parametric) mathematical structure, the security proof of Dilithium only makes use of a select few properties of this structure. For the machine-checked security proof, we insert an extra layer of abstraction. We define an abstract theory defining an abstract ring type R_q together with the various (abstract) supporting algorithms and the properties relating them. We then carry out the entire security proof with respect to these abstract operations. We also prove that the polynomial ring from the specification can be used to implement all operations such that all axioms are satisfied. While this extra layer of abstraction does not remove any proof burden, it allows us to make explicit the minimal structure required to carry out the security proof and cleanly separate the arithmetic reasoning required to build the required structure from the more high-level parts of the security proof.

The supporting algorithms are as follows. In addition to L_1 and the L_∞ norms, written $\|\_\|_1$ and $\|\_\|_\infty$ respectively, we have two rounding functions. Intuitively, $\mathsf{power2round}(r, d)$ rounds to the nearest multiple of 2^d and removes trailing zeros. Similarly, $\mathsf{highBits}(r, \alpha)$ round into α buckets of (roughly) equal size. We treat the result of $\mathsf{highBits}$ as an (abstract) bucket designation while $\mathsf{lowBits}(r, \alpha)$ can be seen as the difference between r and the center of its designated bucket. Lastly, $h := \mathsf{makeHint}(z, r, \alpha)$ creates a "hint" for $\mathsf{useHint}(h, r, \alpha)$ to compute the high bits of $r + z$ without knowing z, provided z is small. All operations, except $\|\_\|_1$ and $\|\_\|_\infty$, are generalized pointwise to vectors R_q^k. The former is only used on R_q while for the latter the vector version is defined as $\|\mathbf{r}\|_\infty := \max_i \|\mathbf{r}_i\|_\infty$. Further, we write S_γ^l for the uniform distribution over R_q^l conditioned on $\|\_\|_\infty \leqslant \gamma$. With the supporting algorithms in place, the Dilithium signature scheme is defined in Fig. 5.

While we present our results using a conventional mathematical presentation, the scheme and the security proof are completely formalized in EasyCrypt (cf. Fig. 13). Note that, in contrast to [KLS18], we are working in a typed setting. In particular, the hash function (or random oracle) H takes pairs (w_1, m), where m is a message and $w : \mathsf{high}_{2\gamma_2}$, as arguments and outputs a uniformly random $c \in B_\tau$:

$$B_\tau := \{c \in R_q \mid \|c\|_\infty = 1 \text{ and } \|c\|_1 = \tau\}$$

In addition to the parameters n and q internal to R_q, the scheme has a number of additional parameters: the size of $\mathbf{A}$ (i.e., $k \times l$) the coefficient ranges for $\mathbf{s}_1, \mathbf{s}_2$ (the interval $[-\eta, \eta]$) and $\mathbf{y}$ (the interval $[-\gamma_1+1, \gamma_1-1]$), the low-order rounding range ($\alpha := 2\gamma_2$), the number d of bits dropped from $\mathbf{t}$, and the number τ of $\pm 1's$ in c (cf. B_τ above). Further, there is the derived parameter $\beta := \tau \cdot \eta$.[11]

We now give some of the properties of R_q and the supporting algorithms that we require for the security proof. Let q and α be integers such that $2\alpha < q$, $q \equiv 1 \mod \alpha$ and α is even. Further let $\mathbf{r}$ and $\mathbf{z}$ be vectors over R_q where $\|\mathbf{z}\|_\infty \leqslant \alpha/2$

[11] See [DKL+21] for a discussion on how these parameters are set in practice.

```
keygen():
1: $\mathbf{A} \leftarrow R_q^{k\times l}$
2: $(\mathbf{s}_1, \mathbf{s}_2) \leftarrow S_\eta^l \times S_\eta^k$
3: $\mathbf{t} := \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
4: $\mathbf{t}_1 := \mathsf{power2round}(\mathbf{t}, d)$
5: $\mathbf{t}_0 := \mathbf{t} - \mathbf{t}_1 \cdot 2^d$
6: $pk := (\mathbf{A}, \mathbf{t}_1)$
7: $sk := (\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$
8: return $(pk, sk)$
```

```
verify(pk, m, σ):
1: $(\mathbf{A}, \mathbf{t}_1) := pk$
2: $(c, (\mathbf{z}, \mathbf{h})) := \sigma$
3: $\mathbf{w}_1 := \mathsf{useHint}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$
4: $c' := H(\mathbf{w}_1, m)$
5: return $[\![\ \|\mathbf{z}\|_\infty < \gamma_1 - \beta \wedge c = c'\ ]\!]$.
```

```
sign(sk, m):
1: $(\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0) := sk$
2: $r := \perp$
3: while $r = \perp$ do
4:     $\mathbf{y} \leftarrow S_{\gamma_1 - 1}^l$
5:     $\mathbf{w} := \mathbf{A}\mathbf{y}$
6:     $w_1 := \mathsf{highBits}(\mathbf{w}, 2\gamma_2)$
7:     $c \in B_\tau := H(w_1, m)$
8:     $z := \mathbf{y} + c\mathbf{s}_1$
9:     if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta \wedge \|\mathsf{lowBits}(\mathbf{w} - c\mathbf{s_2}, 2\gamma_2)\|_\infty < \gamma_2 - \beta$ then
10:        $\mathbf{h} := \mathsf{makeHint}(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$
11:        $r := (\mathbf{z}, \mathbf{h})$
12: return $(c, r)$
```

Fig. 5. The Dilithium signature scheme

and let $\mathbf{h}$ be a vector of hints. We require:

$$\mathsf{useHint}(\mathsf{makeHint}(\mathbf{z}, \mathbf{r}, \alpha), \mathbf{r}, \alpha) = \mathsf{highBits}(\mathbf{r} + \mathbf{z}, \alpha) \tag{17}$$

$$\|\mathbf{r} - \mathsf{shift}_\alpha(\mathsf{useHint}(\mathbf{h}, \mathbf{r}, \alpha))\|_\infty \leqslant \alpha + 1 \tag{18}$$

$$\|\mathbf{r} - \mathsf{power2round}(\mathbf{r}, d) \cdot 2^d\|_\infty \leqslant 2^{d-1} \tag{19}$$

$$\mathsf{shift}_\alpha \text{ is injective} \tag{20}$$

There are, of course, a number of additional properties we require (e.g., $0 \leqslant \|\mathbf{r}\|_\infty$, $\|c\mathbf{t}\|_\infty \leqslant \|c\|_1 \cdot \|\mathbf{t}\|_\infty$, or the triangle inequality $\|\mathbf{u} + \mathbf{v}\|_\infty \leqslant \|\mathbf{u}\|_\infty + \|\mathbf{v}\|_\infty$). For the complete list we refer to the `DRing` theory (for the properties of R_q and the supporting algorithms) and the `DVect` theory for the lifting to vectors.

Even though verify only uses $\mathbf{A}$ and $\mathbf{t}_1$, the security proofs assume that the adversary knows $\mathbf{t}$, allowing it to derive both $\mathbf{t}_1$ and $\mathbf{t}_0$. In particular, the entirety of $\mathbf{t}$ is needed to define the HVZK simulator for the EF-CMA to EF-NMA reduction. Hence, the first step of the proof is to change the public key to $(\mathbf{A}, \mathbf{t})$, the secret key to $(\mathbf{A}, \mathbf{s}_1, \mathbf{s}_2)$, and adapt sign and verify to compute $\mathbf{t}_1$ and $\mathbf{t}_0$ as necessary. We call this scheme *Simplified Dilithium* (DilithiumS) and prove the following lemma showing that it is sufficient to establish security for this variant of the construction.

Lemma 6. *Let $\mathcal{A}^{\mathsf{Sign},H}$ be a CMA attacker against Dilithium. Then there exists an adversary $\mathcal{B}^{\mathsf{Sign},H}$ such that:* $\mathsf{Adv}^{\textit{EF-CMA}}_{\mathsf{Dilithium}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\textit{EF-CMA}}_{\mathsf{DilithiumS}}(\mathcal{B})$ *Further,* $\mathsf{Time}(\mathcal{A}) \approx \mathsf{Time}(\mathcal{B})$.

6.2 Reduction to MLWE and SelfTargetMSIS

We now prove EF-NMA security of the simplified scheme. The reduction to MLWE and SelfTargetMSIS closely follows [KLS18, DKL+21]. We first sketch the mathematical proof for the sake of completeness — we correct minor points wrt the statements in [KLS18] and [DKL+21] that became clear in the formal proof — and then comment on the formalization in EasyCrypt. We begin by recalling the MLWE and SelfTargetMSIS security assumptions for the ring R_q used by Dilithium.

Definition 3 (MLWE Assumption). *Let m and k be integers and let $D : R_q \to [0,1]$ be a distribution. The advantage of an algorithm $\mathcal{A}$ for solving the decisional $\mathsf{MLWE}_{m,k,D}$ problem over the ring R_q is:*

$$\mathsf{Adv}^{\mathsf{MLWE}}_{m,k,D}(\mathcal{A}) := \Big| \Pr\Big[\mathcal{A}(\mathbf{A},\mathbf{t}) = 1 \;\Big|\; \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{t} \leftarrow R_q^m\Big] - \Pr\Big[\mathcal{A}(\mathbf{A},\mathbf{A}\mathbf{s}_1 + \mathbf{s}_2) = 1 \;\Big|\; \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{s}_1 \leftarrow D^k; \mathbf{s}_2 \leftarrow D^m\Big]\Big|$$

Definition 4 (Self-target MSIS Assumption). *Let m and k be integers and let $H : R_q^m \times M \to B_\tau$ be a random oracle.*

$$\mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{H,m,k,\gamma} := \Pr\left[\begin{array}{c} \|\mathbf{r}\|_\infty \leqslant \gamma \\ H([\ \mathbf{I}_m \mid \mathbf{A}\ ] \cdot \mathbf{r}, \mu) = \mathbf{r}[m+k-1] \end{array} \middle| \mathbf{A} \leftarrow R_q^{m\times k}; (\mathbf{r},\mu) \leftarrow \mathcal{A}^H(\mathbf{A})\right]$$

The goal of this section is then to prove the following lemma.

Lemma 7. *For every adversary $\mathcal{A}^{\mathsf{Sign},H}$ breaking NMA security of simplified Dilithium, we can construct an MLWE adversary $\mathcal{B}$ and a SelfTargetMSIS adversary $\mathcal{C}$ such that:*

$$\mathsf{Adv}^{\textit{EF-NMA}}_{\mathsf{DilithiumS}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{k,l,S_\eta}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{G,k,l+1,\zeta}(\mathcal{C})$$

where $\zeta := \max\left\{\gamma_1 - \beta, 2\gamma_2 + 1 + \tau 2^{d-1}\right\}$ and $G : R_q \times \mathsf{Msg} \to B_\tau$ is a random oracle. Further $\mathsf{Time}(\mathcal{A}) \approx \mathsf{Time}(\mathcal{B}) \approx \mathsf{Time}(\mathcal{C})$.

Proof (Sketch). The proof consists of three steps. The first step is to replace keygen with a keygen′, sampling $\mathbf{t}$ uniformly at random and returning an undefined (and unused) secret key (cf. Fig. 6). Taking $\mathcal{B}$ to be the remainder of the EF-NMA security game after key generation, the difference between the two games is exactly $\mathrm{Adv}^{\mathsf{MLWE}}_{k,l,R_q}(\mathcal{B})$. Next, we define an oracle $H'(\mathbf{w}_1, m) := G(\mathsf{shift}_\alpha(\mathbf{w}_1), m)$. Since shift_α is injective and both H (used by our scheme) and G (the random

```
keygen'():
 1: $\mathbf{A} \leftarrow R_q^{k \times l}$
 2: $\mathbf{t} \leftarrow R_q^k$
 3: return $((\mathbf{A}, \mathbf{t}), \mathsf{witness})$
```

```
$\mathcal{C}(\mathbf{A}' : R_q^{k \times l+1})$:
 1: $(\mathbf{A}, \overline{\mathbf{t}}) := \mathbf{A}'$
 2: $\mathbf{t} := -\overline{\mathbf{t}}$
 3: $(m, (c, (\mathbf{z}, \mathbf{h}))) \leftarrow \mathcal{A}^{H'}(\mathbf{A}, \mathbf{t})$
 4: $\mathbf{t}_1 := \mathsf{power2round}(d, \mathbf{t})$
 5: $\mathbf{r} := \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d$
 6: $\mathbf{u}_1 := \mathbf{r} - \mathsf{shift}_\alpha(\mathsf{useHint}(\mathbf{h}, \mathbf{r}, \alpha)$
 7: $\mathbf{u}_2 := c(\mathbf{t} - \mathbf{t}_1 \cdot 2^d)$
 8: return $((\mathbf{u}_1 - \mathbf{u}_2) \,|\, \mathbf{z} \,|\, [c])$
```

Fig. 6. Randomized keygen and reduction to SelfTargetMSIS

oracle from the SelfTargetMSIS assumption) have as output distribution the uniform distribution over B_τ, replacing H by H' incurs no loss.

It remains to construct the reduction $\mathcal{C}$ that returns a valid solution for the SelfTargetMSIS problem whenever $\mathcal{A}^{H'}(\mathbf{A}, \mathbf{t})$ successfully forges a signature (for some $(\mathbf{A}, \mathbf{t})$ derived from the SelfTargetMSIS instance). Writing | for vector concatenation, the reduction is given in Fig. 6. Given a SelfTargetMSIS instance $\mathbf{A}'$ with dimensions $k \times l + 1$, $\mathcal{C}$ splits off the last column, negates it, and passes the parts to the EF-NMA adversary. We have that the distribution of $(\mathbf{A}, \mathbf{t})$ is identical to the EF-NMA game (using keygen' for key generation). Now assume that $(m, (c, (\mathbf{z}, \mathbf{h})))$ passes verification with respect to H'. That is, we have:

1. $H'(\mathsf{useHint}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, \alpha), m) = c$
2. $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$

Now with $\mathbf{r}' := ((\mathbf{u}_1 - \mathbf{u}_2) \,|\, \mathbf{z} \,|\, [c]) \in R_q^{k+l+1}$ as defined in $\mathcal{C}$, we have:

$$\begin{aligned} G([\mathbf{I}_k|\mathbf{A}'] \cdot \mathbf{r}', m) &= G([\mathbf{I}_k|\mathbf{A}| - \mathbf{t}] \cdot \mathbf{r}', m) \\ &= G(\mathbf{A}\mathbf{z} - c\mathbf{t} + (\mathbf{u}_1 - \mathbf{u}_2), m) \\ &= H'(\mathsf{useHint}(\mathbf{h}, \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, \alpha), m) \\ &= \mathbf{r}'[k + l] \end{aligned}$$

Hence, $\mathbf{r}'$ satisfies the "self-target" condition and it remains to show $\|\mathbf{r}'\|_\infty \leqslant \max\left\{\gamma_1 - \beta, 2\gamma_2 + 1 + \tau 2^{d-1}\right\}$. For $\mathbf{z}$ this follows by assumption, and for c we have $\|[c]\|_\infty = 1$. For $(\mathbf{u}_1 - \mathbf{u}_2)$, recalling that we set $\alpha = 2\gamma_2$, we have:

$$\|\mathbf{u}_1 - \mathbf{u}_2\|_\infty \leqslant \|\mathbf{u}_1\|_\infty + \|\mathbf{u}_2\|_\infty \leqslant 2\gamma_2 + 1 + \|c\|_1 \cdot \|\mathbf{t} - \mathbf{t}_1 \cdot 2^d\|_\infty \leqslant 2\gamma_2 + 1 + \tau 2^{d-1}$$

where the bound for $\mathbf{u}_1$ follows with (18) and the bound for $\mathbf{u}_2$ follows with (19). □

The main technical difficulty when formalizing the results in this section was to develop a matrix library that would support all the required operations. This was done in collaboration and is shared between several developments. In the mathematical presentation we have assumed tacitly that all matrix operations are carried out on matrices and vectors of compatible dimensions. In EasyCrypt, we use a matrix theory where operations are defined even if the dimensions do not match, with "undefined" behaviors chosen to simplify the equational theory. This does not cause any problems for matrices and vectors provided by the schemes or games. However, vectors given by the adversary (i.e., the $\mathbf{z}$ and $\mathbf{h}$ component of a signature) need to be checked for the correct length by the verify procedure (cf. Fig 13).

6.3 The HVZK Simulator and EF-CMA Security

We now extend the security proof from EF-NMA to EF-CMA. This mainly amounts to instantiating Theorem 3. In order to do so, we need to express the simplified Dilithium scheme as the FSwA transform of an IDS and provide a HVZK simulator for this IDS. There are two minor technical complications. The first is that, in order to simplify the mechanization of the proof of Theorem 3, we restricted ourselves to IDS where Com, Resp, and Verify were given as operators (i.e., mathematical functions) rather than procedures (i.e., imperative code such as that given in Fig. 5. Now we have to "pay" for this simplification and show that our scheme can indeed be seen as the FS transform of such an operator-based IDS. The second complication is that Dilithium is actually based on a variant of the FS transform that is specific to commitment recoverable IDS, allowing to replace the commitment w with the (in practice much smaller) challenge c in the signature. The difference is mainly in verification as shown below (generic on the left, commitment recoverable on the right):

```
verify($pk, m, \sigma = (w, z)$):
 1: $c := \mathsf{H}(w, m)$
 2: return $[\![\mathsf{Verify}(pk, w, c, z)]\!]$
```

```
verify($pk, m, \sigma = (c, z)$):
 1: $\mathbf{w} := \mathsf{Recover}(pk, c, z)$
 2: return $[\![\mathsf{Verify}(pk, w, c, z)]\!] \wedge [\![c = \mathsf{H}(w, m)]\!]$
```

The Recover function for Dilithium is

$$\mathsf{Recover}((\mathbf{A}, \mathbf{t}), c, (\mathbf{z}, \mathbf{h})) := \mathsf{useHint}(\mathbf{h}, \mathbf{Az} - c \cdot \mathsf{power2round}(\mathbf{t}, d) \cdot 2^d)$$

For Sign (cf. Fig. 5), Lines 4-6 correspond to Com while Lines 8-11 correspond to Resp. Defining the remaining operators and proving that no context can distinguish the original scheme from the FSwA transform of the IDS is routine.

Proving EF-CMA security of the scheme obtained using the FSwA transform for commitment-recoverable IDS can trivially be reduced to proving EF-CMA security of the standard FSwA transform. However, the reduction requires an additional q_S random oracle queries to turn the signatures of the form (w, z), returned by the signing oracle, into signatures of the form (c, z) as expected by the adversary.

Now we define the HVZK Simulator for the IDS sketched above. We let D_z be the distribution that with probability $|S^l_{\gamma_1-\beta-1}|/|S^l_{\gamma_1-1}|$ returns true and otherwise returns false. The Sim in Fig. 7 is a minor variation of the one in [KLS18, Figure 14]. The main difference is that we make explicit the use of Recover to satisfy the interface of Theorem 3. As mentioned earlier, executing Sim in a while loop until $z \neq \perp$ yields an acHVZK simulator.

```
Sim(pk = (A, t)):
1: $r := \bot$
2: $b \leftarrow D_z$
3: if $b$ then
4:     $\mathbf{t}_0 := \mathbf{t} - \mathsf{power2round}(\mathbf{t}, d) \cdot 2^d$
5:     $c \leftarrow B_\tau$
6:     $\mathbf{z} \leftarrow S^l_{\gamma_1-\beta-1}$
7:     if $\|\mathsf{lowBits}(\mathbf{Az} - c\mathbf{t}, \alpha)\|_\infty < \gamma_2 - \beta$ then
8:         $h := \mathsf{makeHint}(-c\mathbf{t}_0, \mathbf{Az} - c\mathbf{t} + c\mathbf{t}_0))$
9:         $r := (\mathsf{Recover}(pk, c, \mathbf{z}), c, (z, h))$
10: return $r$.
```

Fig. 7. HVZK simulator

Putting everything together, we obtain the security theorem for Dilithium as we have formalized it in EasyCrypt:

Theorem 4. *Let Γ, δ, ϵ and $p_0 < 1$ be such that* $\Pr_{A \leftarrow R_q^{k \times l}}[\neg\Gamma] \leqslant \delta$,

$$\mathbb{E}_{A \leftarrow R_q^{k \times l}}\left[\max_w \Pr_{\mathbf{y} \leftarrow S^l_{\gamma_1 - 1}}[\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2) = w] \;\middle|\; \Gamma\right] \leqslant \epsilon, \text{ and}$$

$\Pr_{\mathbf{z} \leftarrow S^l_{\gamma_1-\beta-1}}[\|\mathsf{lowBits}(\mathbf{Az} - c\mathbf{t}, 2\gamma_2)\|_\infty \geqslant \gamma_2 - \beta] \leqslant p_0$ *for all* $\mathbf{A}$ *satisfying* Γ*, all* $c \in B_\tau$ *and all* $\mathbf{t} \in R_q^l$*. Then for every classical adversary* $\mathcal{A}^{\mathsf{Sign},H}$ *making at most* q_S *signing queries and at most* q_H *random oracle queries and for the adversaries* $\mathcal{B}$ *(against* MLWE*) and the* $\mathcal{C}^G$ *(against* SelfTargetMSIS*) constructed in the proof of Lemma 7 we have:*

$$\mathsf{Adv}^{\textit{EF-CMA}}_{\mathsf{Dilithium}}(\mathcal{A}) \leqslant \mathsf{Adv}^{\mathsf{MLWE}}_{k,l,S_\eta}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{SelfTargetMSIS}}_{G,k,l+1,\zeta}(\mathcal{C})$$
$$\frac{2q_S(q_H + q_S + 1)\epsilon}{1-p} + \frac{q_S\epsilon(q_S+1)}{2(1-p)^2} + \delta$$

where $p := \frac{|S^l_{\gamma_1-\beta-1}|}{|S^l_{\gamma_1-1}|}p_0 + \left(1 - \frac{|S^l_{\gamma_1-\beta-1}|}{|S^l_{\gamma_1-1}|}\right)$ *and* $\zeta := \max\{\gamma_1 - \beta, 2\gamma_2 + 1 + \tau 2^{d-1}\}$.

We remark that we do not show in EasyCrypt that the execution times of $\mathcal{B}$ and $\mathcal{C}$ are close to the execution time of $\mathcal{A}$, but this can be checked by inspection. As a consequence, the formal statement needs to be with respect to specific reductions rather than existentially quantified adversaries (cf. Fig. 14).

7 Concrete Security Analysis

In the following, we quantify the security loss of our proof to analyze the impact on the concrete security of Dilithium. Our proof of security for Dilithium has the same overall structure as the one given in [KLS18] regarding the reductions from the underlying computational assumptions MLWE and SelfTargetMSIS. Indeed, the bounds we establish for the advantage of both classical and quantum attackers differ from the original proofs only in the additive terms, which in our case are larger due to additional (Q)ROM reprogramming steps.

In the NIST submission [DKL+21, Section 6.2] the authors simplify the additive security loss as 2^{-254} — a conservative value — and claim that this bound is achieved for all of the parameter sets considered, based on the analysis performed in [KLS18]. In what follows, we give more precise bounds for this additive loss according to our corrected proofs. We show that it is still low enough to comfortably meet the requirements of the relevant NIST security levels.

We recall the expressions for the security loss L in the ROM from Theorem 4, and its quantum counter part L^* obtained from Theorem 2, i.e.

$$L := \frac{2q_S(q_H + q_S + 1)\epsilon}{1-p} + \frac{q_S\epsilon(q_S+1)}{2(1-p)^2} + \delta$$

$$L^* := \frac{2q_S\sqrt{\epsilon}}{1-p}\sqrt{q_H + 1 + \frac{q_S}{1-p}} + 2(q_H+1)\sqrt{\frac{q_S\epsilon}{1-p}} + \delta$$

We present an extended analysis of the bounds on ϵ and δ for the different parameter settings for Dilithium (for the different NIST levels) in Appendix A, using a computer-aided analysis of the distribution of the rank of (the upper square part of) the matrix $\mathbf{A}$. We note that δ and ϵ are related and allow for different tradeoffs for fixed parameters which we did not fully exploit, yet. For the rejection probability p, we use the heuristic from [KLS18] to treat $\mathsf{lowBits}(\mathbf{Az} - c\mathbf{t})$ as uniformly random in $S^k_{\gamma_2-1}$. This gives rise to the following table.

	p	q_S	q_H	δ	ϵ	loss
NIST2	$\leqslant \frac{49}{64}$	2^{64}	2^{128}	2^{-209}	2^{-403}	$L \leqslant 2^{-206}$
				2^{-64}	2^{-446}	$L^* \leqslant 2^{-58}$
			2^{64}	2^{-265}	2^{-390}	$L \leqslant 2^{-257}$
				2^{-117}	2^{-428}	$L^* \leqslant 2^{-113}$
			1	2^{-265}	2^{-390}	$L \leqslant 2^{-257}$
				2^{-117}	2^{-428}	$L^* \leqslant 2^{-113}$
NIST3	$\leqslant \frac{103}{128}$	2^{64}	2^{192}	2^{-867}	2^{-1108}	$L \leqslant 2^{-847}$
				2^{-362}	2^{-1180}	$L^* \leqslant 2^{-360}$
NIST5	$\leqslant \frac{759}{1024}$	2^{64}	2^{256}	2^{-1268}	2^{-1584}	$L \leqslant 2^{-1260}$
				2^{-540}	2^{-1664}	$L^* \leqslant 2^{-538}$

Fig. 8. Concrete security loss of Dilithium from Theorems 2 (L^*) and 4 (L).

The take-away from our analysis is that the statistical additive loss remains sufficiently small for all scenarios and therefore the dominant terms for the security level will remain the bounds for MLWE and SelfTargetMSIS. To be more precise, the table says that an attacker doing q_H hash computations only gains an additive advantage of 2^{-58} in the worst case (quantum attack against level 2). For reference, NIST security level 2 corresponds to a setting where the expected cost of a successful attack should match that of a collision search in a generic 256-bit hash function. This is often estimated to be $256/3 \approx 86$. So, after 2^{86} quantum queries, we would expect to find a collision. In our case, even after 2^{128} quantum queries, the success probability is bounded by 2^{-58}. Actually, one number that may appear debatable (in the sense of really guaranteeing the claimed security) is the bound for level 2 after a single query of a success probability of 2^{-113}. This number is caused by the number of signing queries which dominates in this case. This implies that for this attack, the cost is also dominated by the signing queries (here 2^{64}). What the number says is that, if one could ignore the cost of the signing queries, then there would exist an attack with an expected cost of about 2^{113} which is just the number of hash queries. However, given that the cost of each of these attacks is at least 2^{64} the total attack cost is 2^{177}. Hence, for all the parameters there is a comfortable margin regarding the security loss induced by the reduction. Thereby the full security of Dilithium is still determined by the hardness of solving MLWE and $\mathsf{SelfTargetMSIS}$.

8 Acknowledgments

Jelle Don is supported by the ERC-ADG project ALGSTRONGCRYPTO (Project No. 740972). Benjamin Grégoire is supported by the Agence Nationale de la Recherche (French National Research Agency) as part of the France 2030 programme - ANR-22-PECY-0006. Yu-Hsuan Huang is supported by the Dutch Research Agenda (NWA) project HAPKIDO (Project No. NWA.1215.18.002), which is financed by the Dutch Research Council (NWO). Andreas Hülsing is supported by an NWO VIDI grant (Project No. VI.Vidi.193.066). Xiaodi Wu is supported by AFOSR Young Investigator Program (YIP) Award (FA95502110094) and NSF CAREER Award (NSF-CCF-1942837).

References

ABG$^+$23. Martin Avanzini, Gilles Barthe, Benjamin Grégoire, Georg Moser, and Gabriele Vanoni. A mechanisation of the complexity analysis of skiplists. Unpublished manuscript, 2023.

BBB$^+$21. Manuel Barbosa, Gilles Barthe, Karthik Bhargavan, Bruno Blanchet, Cas Cremers, Kevin Liao, and Bryan Parno. Sok: Computer-aided cryptography. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 777–795. IEEE, 2021.

BBF$^+$21. Manuel Barbosa, Gilles Barthe, Xiong Fan, Benjamin Grégoire, Shih-Han Hung, Jonathan Katz, Pierre-Yves Strub, Xiaodi Wu, and Li Zhou. Easypqc: Verifying post-quantum cryptography. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2564–2586, New York, NY, USA, 2021. Association for Computing Machinery.

BBK17. Karthikeyan Bhargavan, Bruno Blanchet, and Nadim Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In *IEEE Symposium on Security and Privacy (S&P)*, pages 483–502. IEEE Computer Society, 2017.

BDK$^+$22. Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 95–126. Springer, 2022.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 464–492. Springer, 2020.

CHH$^+$17. Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1773–1788. ACM, 2017.

CHSvdM16. Cas Cremers, Marko Horvat, Sam Scott, and Thyla van der Merwe. Automated analysis and verification of TLS 1.3: 0-rtt, resumption and delayed authentication. In *IEEE Symposium on Security and Privacy (S&P)*, pages 470–485. IEEE Computer Society, 2016.

DFG19. Luca De Feo and Steven D Galbraith. Seasign: compact isogeny signatures from class group actions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 759–789. Springer, 2019.

DFK$^+$17. Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Jonathan Protzenko, Aseem Rastogi, Nikhil Swamy, Santiago Zanella Béguelin, Karthikeyan Bhargavan, Jianyang Pan, and Jean Karim Zinzindohoue. Implementing and proving the TLS 1.3 record layer. In *IEEE Symposium on Security and Privacy (S&P)*, pages 463–482. IEEE Computer Society, 2017.

DFPS23. Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A Detailed Analysis of Fiat-Shamir with Aborts. *Cryptology ePrint Archive*, 2023.

DKL$^+$18. Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

DKL$^+$21. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, , and Damien Stehlé. Crystals-dilithium – algorithm specifications and supporting documentation (version 3.1). Technical report, February 2021. Specification document.

GHHM21. Alex B Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Tight adaptive reprogramming in the qrom. In *Advances in Cryptology – ASIACRYPT 2021*, pages 637–667. Springer, 2021.

KLS18. Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 552–586, Cham, 2018. Springer International Publishing.

Koz83. Dexter Kozen. A probabilistic pdl. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, page 291–297, New York, NY, USA, 1983. Association for Computing Machinery.

LNP22. Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plancon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. *Cryptology ePrint Archive*, 2022.

Lyu09. Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer, 2009.

Lyu12. Vadim Lyubashevsky. Lattice signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 738–755. Springer, 2012.

Zha19. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 239–268. Springer, 2019.

A Bounding the Min-Entropy

Below, we show, partly computer aided, a lower bound on the min-entropy, i.e., an upper bound on the guessing probability, of the first message $\mathsf{highBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$ in the IDS underlying Dilithium for some relevant choices of the parameters.

In more detail, as specified in Dilithium, let $R_q \cong \mathbb{F}_q[X]/(X^n+1)$ where X^n+1 splits completely in $\mathbb{F}_q$, i.e. there exist pair-wise distinct $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$ such that $X^n+1 = (X-\alpha_1)\cdots(X-\alpha_n)$, and let $k \geqslant l$. Then, we will show that, with overwhelming probability over the choice of $\mathbf{A} \leftarrow R_q^{k\times l}$, the min-entropy of $\mathsf{highBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$ is large over the choice of $\mathbf{y} \leftarrow S^l_{\gamma_1-1}$. We denote $H_{\min}(\mathbf{x})$ as the min-entropy of a random variable $\mathbf{x}$.

A.1 Controlling the Min-Entropy via the Rank of A

First, note that, for the top-most square $\mathbf{A}^\square \in R_q^{l\times l}$ of $\mathbf{A} \in R_q^{k\times l}$,

$$H_{\min}(\mathsf{highBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)) \geqslant H_{\min}(\mathsf{highBits}(\mathbf{A}^\square\mathbf{y}, 2\gamma_2)) \geqslant H_{\min}(\mathbf{A}^\square\mathbf{y}) - nl\lg(2\gamma_2+1)\,.$$

Furthermore,

$$H_{\min}(\mathbf{A}^\square\mathbf{y}) \geqslant H_{\min}(\mathbf{y}) - (nl - \mathsf{rank}(\mathbf{A}^\square))\lg(q)\,,$$

where $\mathsf{rank}(\mathbf{A}^\square)$ is the rank of $\mathbf{A}^\square$ acting on R_q^l as a $\mathbb{F}_q$-linear space. Finally, by the choice of $\mathbf{y}$, $H_{\min}(\mathbf{y}) \geqslant nl\lg(2\gamma_1-1)$. Thus, altogether,

$$H_{\min}(\mathsf{highBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)) \geqslant nl\lg\left(\frac{2\gamma_1-1}{2\gamma_2+1}\right) - (nl - \mathsf{rank}(\mathbf{A}^\square))\lg(q)\,. \quad (21)$$

Therefore, it suffices to have good enough control over $\mathsf{rank}(\mathbf{A}^\square)$.

A.2 Controlling the Rank of A

The following is a consequence of the requirement that $X^n + 1$ splits completely in $\mathbb{F}_q$.

Lemma 8. *There is a $\mathbb{F}_q$-algebra isomorphism between*

$$\phi : R_q^{l\times l} \xrightarrow{\sim} \bigoplus_{1\leqslant i\leqslant n} \mathbb{F}_q^{l\times l} .$$

Furthermore, ϕ is $\mathbb{F}_q$-rank-preserving, i.e. for every $\mathbf{A}^{\square}$, with $\phi(\mathbf{A}^{\square}) = \bigoplus_i \mathbf{D}_i$, we have $\mathsf{rank}(\mathbf{A}^{\square}) = \mathsf{rank}(\phi(\mathbf{A}^{\square})) = \sum_i \mathsf{rank}(\mathbf{D}_i)$.

From Lemma 8, we now know that the distribution of $\mathsf{rank}(\mathbf{A}^{\square})$ equals the distribution of $\sum_i \mathsf{rank}(\mathbf{D}_i)$ for random and independent $\mathbf{D}_1, \ldots, \mathbf{D}_n \leftarrow \mathbb{F}_q^{l\times l}$.

The Rank of a Random Matrix Below, we thus consider a uniformly random $\mathbf{D} \leftarrow \mathbb{F}_q^{l\times l}$, and we work out the distribution of $\mathsf{rank}(\mathbf{D})$. For this purpose, let

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}^1 & \ldots & \mathbf{D}^l \end{pmatrix} ,$$

where each $\mathbf{D}^j \in \mathbb{F}_q^l$ is the jth column of $\mathbf{D}$. Define the *rank sequence* $r_1, \ldots, r_l$ given by

$$r_j := \mathsf{rank} \begin{pmatrix} \mathbf{D}^1 & \ldots & \mathbf{D}^j \end{pmatrix} .$$

With the convention that $r_0 := 0$, define their difference sequence $d_1, \ldots, d_l$ as $d_j := r_j - r_{j-1} \in \{0,1\}$. In other words, d_j indicates whether the j-th column increases the rank or not.

Lemma 9 below gives the distribution of the difference sequence $(d_1, \ldots, d_n)$ for a random $\mathbf{D}$.

Lemma 9. *The probability that a random matrix $\mathbf{D} \in \mathbb{F}_q^{l\times l}$ has a given difference sequence $(d_1, \ldots, d_l) \in \{0,1\}^l$ is*

$$\pi_l(q, d_1, \ldots, d_l) := \prod_{1\leqslant j\leqslant l} \left(d_j + (-1)^{d_j} q^{-(l-r_{j-1})} \right) ,$$

where $r_1, \ldots, r_l$ is naturally defined as $r_j = d_1 + \cdots + d_j$, with $r_0 = 0$.

Proof. For any $j \in \{1, \ldots, l\}$, conditioned on the columns $\mathbf{D}^1, \ldots, \mathbf{D}^{j-1}$, the matrix $\mathbf{D}$ has $d_j = 0$ if and only if the j-th column $\mathbf{D}^j$ lies in $\mathsf{span}_{\mathbb{F}_q}\{\mathbf{D}^1, \ldots, \mathbf{D}^{j-1}\}$, which happens with probability

$$\Pr\left[\mathbf{D}^j \in \mathsf{span}_{\mathbb{F}_q}\{\mathbf{D}^1, \ldots, \mathbf{D}^{j-1}\}\right] = \frac{\#\mathsf{span}_{\mathbb{F}_q}\{\mathbf{D}^1, \ldots, \mathbf{D}^{j-1}\}}{q^l} = q^{-(l-r_{j-1})} ,$$

and it has $d_j = 1$ with complementary probability $1 - q^{-(l-r_{j-1})}$. The probability of a particular difference sequence $d_1, \ldots, d_l$ is then the product of the respective probabilities above, which matches the claim. $\square$

Since $\mathsf{rank}(\mathbf{D}) = d_1 + \cdots + d_l$, we have

$$\Pr[\mathsf{rank}(\mathbf{D}) = r] = \sum_{d \in S_r^l} \pi_l(q, d) \ , \tag{22}$$

where $S_r^l := \{(d_1, \ldots, d_l) \in \{0,1\}^l \mid d_1 + \cdots + d_l = r\}$. Note that, by using Lemma 9, one can show that $\Pr[\mathsf{rank}(\mathbf{D}) = r] = p_r(1/q)$ for a (l-dependent) integer polynomial p_r of degree at most l^2.

The equality (22) gives rise to an algorithm that computes the distribution of $\Pr[\mathsf{rank}(\mathbf{D}) = r]$ (as polynomials in $1/q$) in time $2^{O(l)}$. For $l = 5$ (which is in line with the NIST3 parameters of Dilithium) one obtains the polynomials given in Fig. 9.[12]

$$\begin{aligned}
p_0(1/q) &= q^{-25}\\
p_1(1/q) &= -q^{-25} - q^{-24} - q^{-23} - q^{-22} - q^{-21} + q^{-20} + q^{-19} + q^{-18} + q^{-17} + q^{-16}\\
p_2(1/q) &= q^{-24} + q^{-23} + 2q^{-22} + 2q^{-21} + q^{-20} - q^{-19} - 2q^{-18} - 4q^{-17} - 4q^{-16}\\
&\quad - 2q^{-15} - q^{-14} + q^{-13} + 2q^{-12} + 2q^{-11} + q^{-10} + q^{-9}\\
p_3(1/q) &= -q^{-22} - q^{-21} - 2q^{-20} - q^{-19} + 3q^{-17} + 4q^{-16} + 5q^{-15} + 3q^{-14}\\
&\quad - 3q^{-12} - 5q^{-11} - 4q^{-10} - 3q^{-9} + q^{-7} + 2q^{-6} + q^{-5} + q^{-4}\\
p_4(1/q) &= q^{-19} + q^{-18} - q^{-16} - 2q^{-15} - 3q^{-14} - 2q^{-13} + q^{-12} + 3q^{-11} + 4q^{-10}\\
&\quad + 3q^{-9} + q^{-8} - 2q^{-7} - 3q^{-6} - 2q^{-5} - q^{-4} + q^{-2} + q^{-1}\\
p_5(1/q) &= -q^{-15} + q^{-14} + q^{-13} - q^{-10} - q^{-9} - q^{-8} + q^{-7} + q^{-6} + q^{-5} - q^{-2} - q^{-1} + 1 \ .
\end{aligned}$$

Fig. 9. The polynomials $p_r(1/q) = \Pr[\mathsf{rank}(\mathbf{D}) = r]$ for $l = 5$.

The Rank of a Random Block-Diagonal Matrix Towards controlling the distribution of $\mathsf{rank}(\mathbf{A}^{\square})$, we consider the generating function for the distribution of $\mathsf{rank}(\mathbf{D})$, given by

$$f(z) := \sum_{0 \leqslant r \leqslant l} \Pr[\mathsf{rank}(\mathbf{D}) = r] \cdot z^r = \sum_{0 \leqslant r \leqslant l} p_r(1/q) \cdot z^r \ . \tag{23}$$

Then, the nth power $f^n(z)$ of the above generating function generates the distribution of $\mathsf{rank}(\mathbf{A}^{\square}) = \mathsf{rank}(\mathbf{D}_1) + \cdots + \mathsf{rank}(\mathbf{D}_n)$, i.e.

$$f^n(z) = \sum_{0 \leqslant r \leqslant ln} \Pr[\mathsf{rank}(\mathbf{A}^{\square}) = r] \cdot z^r \ .$$

[12] For such a small choice of l the exponential run time is no issue. As a matter of fact, this could still be worked out by hand.

On input $f(z)$, as a polynomial in z with degree l, with coefficient being polynomials in $1/q$ with degree (at most) l^2, the n-th power $f^n(z)$ can be computed in time polynomial in n and l. $\Pr[\mathsf{rank}(\mathbf{A}^{\square}) = r]$ can then be obtained by reading out the coefficient of the degree-r term of $f^n(z)$, which is again an integer polynomial in $1/q$, and evaluating it (as a rational number) for the considered choice of $q \in \mathbb{Z}$.

For any bound B, we can then compute

$$\Pr[\mathsf{rank}(\mathbf{A}^{\square}) < B] = \sum_{r<B} \Pr[\mathsf{rank}(\mathbf{A}^{\square}) = r]$$

as a rational number $\frac{a}{b}$ with $a \leqslant b \in \mathbb{Z}$, which we can then upper bound by writing $b = ad + e$ for $e \in \{0, \ldots, a-1\}$, and noting that

$$\Pr[\mathsf{rank}(\mathbf{A}^{\square}) < B] = \frac{a}{b} = \frac{a}{ad+e} \leqslant \frac{1}{d}.$$

Finally, counting the number of bits in the bit representation of d excluding the most significant bit gives us $\delta \in \mathbb{Z}$ with $2^{\delta} \leqslant d$, and thus $\Pr[\mathsf{rank}(\mathbf{A}^{\square}) < B] \leqslant 2^{-\delta}$.

A.3 Plugging in the Numbers

	n	ℓ	q	γ_1	γ_2
NIST2	256	4	8380417	2^{17}	$(q-1)/88$
NIST3	256	5	8380417	2^{19}	$(q-1)/32$
NIST5	256	7	8380417	2^{19}	$(q-1)/32$

Fig. 10. NISTι parameters for Dilithium, $\iota \in \{2, 3, 5\}$

For the NIST3 parameters as described in Fig. 10, we present the relevant quantities, obtained by following the above computation steps using Sage. Recall that the rank distribution of D, and thus the generating function $f(z)$, is already given above in Fig. 9. In Fig. 11 below are the obtained upper bounds δ_i such that $\Pr[\mathsf{rank}(\mathbf{A}^{\square}) < nl - i] \leqslant \delta_i$ for selective choices of i's, together with the resulting bounds $H_{\min}(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)) \geqslant \eta_i$ obtained via (21).

In particular, we see that except with probability at most 2^{-209} the matrix $\mathbf{A}^{\square}$ has corank at most 11, and then

$$H_{\min}(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)) \geqslant 1028,$$

which means that the guessing probability is at most

$$\text{Guess}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big) \leqslant 2^{-1028},$$

i	0	1	2	3	4	5	6	7	8	9	10	11	19	28	44	63
$-\lg \delta_i$	14	31	47	64	81	99	117	135	153	172	190	209	362	540	867	1268
$\eta_i^{(2)}$	471	448	425	402	379	356	333	310	287	264	241	218	34	0	0	0
$\eta_i^{(3)}$	1281	1258	1235	1212	1189	1166	1143	1120	1097	1074	1051	1028	844	637	269	0
$\eta_i^{(5)}$	1794	1771	1748	1725	1702	1679	1656	1633	1610	1587	1564	1541	1357	1150	782	345

Fig. 11. $\Pr\left[H_{\min}(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)) \geqslant \eta_i^{(\iota)}\right] \leqslant \delta_i$ for NISTι parameters, $\iota \in \{2, 3, 5\}$

where the randomness is over the choice of $\mathbf{y}$.

We can obtain a slightly better bound by considering the *average* guessing probability over the choice of $\mathbf{A}$, averaged over the non-normalized distribution of $\mathbf{A}$ conditioned on $\mathbf{A}^{\square}$ having corank at most 11. Concretely, letting Γ_{11} be the event that $\mathbf{A}^{\square}$ has corank at most 11, where we know that $\Pr[\neg\Gamma_{11}] \leqslant 2^{-209}$, we obtain

$$\begin{aligned}
&\mathbb{E}_A[\mathrm{Guess}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big) \mid \Gamma_{11}] \\
&= \mathbb{E}_A\big[\mathrm{Guess}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big) \mid \mathsf{rank}(\mathbf{A}^{\square}) \geqslant nl - 11\big] \\
&\leqslant \sum_{0 \leqslant i \leqslant 11} \Pr[\mathsf{rank}(\mathbf{A}^{\square}) = nl - i \mid \Gamma_{11}] \cdot 2^{-\eta_i^{(3)}} \leqslant \sum_{0 \leqslant i \leqslant 11} \frac{\delta_{i-1}}{\Pr[\Gamma_{11}]} \cdot 2^{-\eta_i^{(3)}} \leqslant 2^{-1213} ,
\end{aligned}$$

with the convention that $\delta_{-1} = 1$.

Similarly, one can work out the entropy bounds for other NISTι parameters where $\iota \in \{2, 3, 5\}$. In Fig. 12, for each event Γ_I that $\mathbf{A}^{\square}$ has corank at most I we provide an upperbound for $\Pr[\neg\Gamma_I]$, a lowerbound for $H_{\min}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big)$ conditiond on Γ_I happening, and an upperbound on $\mathbb{E}_A[\mathrm{Guess}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big) \mid \Gamma_I]$.

ι	I	$\Pr[\neg\Gamma_I]$	$H_{\min}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big)$	$\mathbb{E}_A[\mathrm{Guess}\big(\mathsf{highBits}(\mathbf{Ay}, 2\gamma_2)\big) \mid \Gamma_I]$
2	3	$\leqslant 2^{-64}$	$\geqslant 402$	$\leqslant 2^{-446}$
	11	$\leqslant 2^{-209}$	$\geqslant 218$	$\leqslant 2^{-403}$
3	19	$\leqslant 2^{-362}$	$\geqslant 844$	$\leqslant 2^{-1180}$
	44	$\leqslant 2^{-867}$	$\geqslant 269$	$\leqslant 2^{-1108}$
5	28	$\leqslant 2^{-540}$	$\geqslant 1150$	$\leqslant 2^{-1664}$
	63	$\leqslant 2^{-1268}$	$\geqslant 345$	$\leqslant 2^{-1584}$

Fig. 12. Entropy bounds for NISTι parameters, $\iota \in \{2, 3, 5\}$

B EasyCrypt Code

Here we give two examples of code from the formal development accompanying this paper. Fig. 13 shows how we formalize the Dilithium signature scheme as a module in EasyCrypt. The scheme is parameterized by a hash function which will be provided by a random oracle in the security proof. Fig. 14 shows the formalization of the statement of Theorem 4. We only show the most important assumptions, i.e., those dealing with query bounds, entropy, and abort probability. For the remaining assumptions (e.g., the conditions on the parameters for the scheme) see the formal development.

```
module type Hash = {
  proc get(x : high list * M) : challenge_t
}.

module Dilithium (H: Hash) = {
  proc keygen() : PK * SK = {
    mA <$ dmatrix dRq k l;
    s1 <$ dvector (dRq_ eta_) l;
    s2 <$ dvector (dRq_ eta_) k;
    t  <- mA *^ s1 + s2;
    t1 <- base2highbitsV t;
    t0 <- base2lowbitsV t;
    pk <- (mA, t1);
    sk <- (mA, s1, s2, t0);
    return (pk, sk);
  }

  proc sign(sk: SK, m: M) : Sig = {
    (mA, s1, s2, t0) <- sk;
    response <- None;

    while(response = None) {
      y <$ dvector (dRq_ (gamma1 - 1)) l;
      w <- mA *^ y;
      w1 <- highBitsV w;
      c <- H.get((w1, m));
      z <- y + c ** s1;
      if(inf_normv z < gamma1 - beta_ /\
         inf_normv (lowBitsV (mA *^ y - c ** s2)) < gamma2 - beta_) {
        h <- makeHintV (- c ** t0) (w - c ** s2 + c ** t0);
        response <- Some(z,h);
      }
    }
    return (c, oget response);
  }

  proc verify(pk: PK, m : M, sig : Sig) = {
    (mA, t1) <- pk;

    (c, response) <- sig;
    (z, h) <- response;
    w1 <- useHintV h (mA *^ z - c ** base2shiftV t1);
    c' <- H.get((w1, m));
    return size z = l /\ size h = k /\ inf_normv z < gamma1 - beta_ /\ c = c';
  }
}.
```

We write base2highbits for power2round and base2lowbits for $\mathbf{t} - \mathsf{power2round}(\mathbf{t}, d) \cdot 2^d$. Further, we omit the fixed arguments (i.e., $2\gamma_2$ for highBits, lowBits, etc. and d for the aforementioned base2 operations).

Fig. 13. Specification of Dilitihium in EasyCrypt.

```
(*** Query Bounds ***)
const qS : { int | 0 ⩽ qS } as qS_ge0. (* number of queries to "sign"  *)
const qH : { int | 0 ⩽ qH } as qH_ge0. (* number of queries to "H.get" *)

(*** Entropy Bounds ***)
(* Abbreviations for the way A, y, and z are sampled *)
op dA = dmatrix dRq k l.
op dy = dvector (dRq_ (gamma1 − 1)) l.
op dz = dvector (dRq_ (gamma1 − beta_ − 1)) l.

(* The check on the matrix component of the keys                   *)
(* This is just predicate used in the proofs, we never compute with it  *)
op check_mx : matrix → bool.

(* upper bound on the mass of the matrices not passing check            *)
const delta_ : { real | 0%r ⩽ delta_ }  as delta_gt0.
axiom check_mx_most : mu dA (predC check_mx) ⩽ delta_.

(* upper bound on the expected probability mass
   of the most likely commitment for a good key  *)
const eps_comm  : { real | 0%r < eps_comm } as eps_comm_gt0.
axiom check_mx_entropy :
  E (dcond dA check_mx) (fun mA ⇒
    p_max (dmap dy (fun y ⇒  highBitsV (mA *^ y)))) ⩽ eps_comm.

(* bound on the probability that he low−bis check in the Sim fails *)
const eps_low : { real | eps_low < 1%r } as eps_low_lt1.
axiom bound_low c (t : vector) (mA : matrix) :
  c ∈ dC τ ⇒  t ∈ dvector dRq k ⇒  check_mx mA ⇒
  mu dz (fun z ⇒  gamma2 − beta_ ⩽ inf_normv (lowBitsV (mA *^ z − c ** t)) ) ⩽ eps_low.

section PROOF.
(* We assume some adversary A *)
declare module A <: Adv_EFCMA_RO{ ··· }
(* A makes no more than qH random oracle (i.e., hash) queries
  and no more than qS many signing queries *)
declare axiom A_bound
  (H' <: Hash{−SD.CountS, −SD.CountH, −A} )
  (SO' <: SOracle_CMA{−SD.CountS, −SD.CountH, −A} ) :
  hoare[ A(SD.CountH(H'), SD.CountS(SO')).forge :
      SD.CountH.qh = 0 ∧ SD.CountS.qs = 0 ⟹  SD.CountH.qh ⩽ qH ∧ SD.CountS.qs ⩽ qS].

op p0 = (size (to_seq (support dz)))%r / (size (to_seq (support dy)))%r.
op p_rej : real = (p0 * eps_low) + (1.0 − p0).

lemma Dilithium_secure &m :
 Pr[EF_CMA_RO(Dilithium, A, H, O_CMA_Default).main() @ &m : res] ⩽
    `|Pr[MLWE_L(RedMLWE(A)).main() @ &m : res] −
      Pr[MLWE_R(RedMLWE(A)).main() @ &m : res]| +
    Pr[SelfTargetMSIS(RedStMSIS(A), SD.RqStMSIS.PRO.RO).main () @ &m : res] +
    (2%r * qS%r * (qH + qS + 1)%r * eps_comm / (1%r − p_rej) +
     qS%r * eps_comm * (qS%r + 1%r) / (2%r * (1%r − p_rej) ^ 2)) + delta_.

end section PROOF.
```

Fig. 14. The formalized security statement (with the most important assumptions)