Multi-Splitting Forking Based Modular Security of Signatures in Multivariate Quadratic Setting

Sanjit Chatterjee[1], Tapas Pandit[2], and Subhabrata Samajder[3]

[1]Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India., E-mail: sanjit@iisc.ac.in
[2]Plaksha University, Mohali, India., E-mail: tapas.pandit@plaksha.edu.in
[3]Institute of Advancing Intelligence, TCG Centers for Research and Education in Science and Technology, Kolkata, India., E-mail: subhabrata.samajder@tcgcrest.org

Abstract

This paper proposes modular security proofs for some identification scheme (IDS)-based signature schemes in the multivariate quadratic (MQ) setting. More precisely, our contributions include concrete security reduction for both 3-pass and 5-pass MQDSS signature schemes in the random oracle model. Although no formal security argument for the former was available in the literature, the one for the latter provides only a qualitative treatment. Our concrete analysis shows that the 3-pass scheme enjoys a comparatively tighter reduction. This result, considered in conjunction with a reported attack on the 5-pass MQDSS from the NIST PQC competition, thus indicates that contrary to the initial suggestion, the 3-pass MQDSS could be a better choice at a concrete security level. Our next focus is on the only blind signature scheme available in the MQ-setting, proposed by Petzoldt et al. While the security of the original scheme was discussed in a non-standard and significantly weak model; we propose a concrete security reduction for a slightly modified scheme in the standard one-more unforgeability (OMF) model.

Central to all our modular proofs are new forking algorithms. The forking algorithm/lemma has been widely used in the formal security reduction of numerous cryptographic schemes, mainly in the discrete logarithm and RSA setting. The abstractions proposed here allow multiple forkings at the same index while satisfying certain additional conditions for the underlying IDS in the MQ-setting. Thus, the forking algorithms capture the nuances of the MQ-setting while being agnostic of the underlying structure.

Keywords: One-more-unforgeability, Existential unforgeability, Blind signature, Multivariate Cryptography, Post-Quantum Security, Splitting Lemma, Forking Algorithm

1 Introduction

Digital signature is a fundamental cryptographic primitive which in its basic version provides authenticity and non-repudiation service. In addition, specialised variants, like blind signature, id-based signature, ring signature, group signature, proxy signature, etc. are available in the literature catering to myriad applications. Blind signature, in particular, is a powerful primitive due to its application as a primary building block in several privacy-preserving protocols, like e-Cash [Cha82, CHL05], e-Voting [OAR$^+$07], anonymous credentials [CL01], etc. Chaum [Cha82] introduced the concept of blind signature that allows a user to obtain a signature for a (blind) message through some interactions with a signing authority. In the literature, many blind signature schemes have been studied of which several [Oka92, Oka92] are built on top of some identification scheme (IDS) like [Sch89].

IDS based (blind) signature schemes alluded to above are in the RSA/discrete-logarithm (DL) setting and typically use rewinding or forking abstractions [PS96b, BN06, HKL19] in the security reduction. However, cryptographic schemes based on DLP or factorisation are known to be insecure in the presence of a quantum adversary. Hence, there is a need for post-quantum alternatives that run on classical hardware but are expected to be secure against both classical and quantum computers. For digital signature, several potential post-quantum candidates are available in the non-linear setting of multivariate quadratic polynomials (MQ). For example, two signature schemes in the MQ setting, Rainbow [DS05] and GeMSS [NIS19] were shortlisted for the third round of evaluation in the NIST PQC standardisation competition[NIS20]; though none of them was finally chosen. Afterwords, NIST initiated a separate competition for additional post-quantum signatures [NIS23], focussing on two aspects: short signature and fast verification. Several proposals in the MQ setting, namely MAYO [Beu21], QR-UOV [FIH$^+$23], SNOVA [WCD$^+$24], UOV [KPG99] as well as one in the MPC-in-the-head setting, namely MQOM [BFR23] are currently under consideration in the second round of these additional signatures [NIS25]. Efficiency in terms of signature size, signing, and verification time makes MQ-based signatures a popular choice for practical applications. However, they suffer from a relatively larger key size.

Amongst the schemes mentioned above, MQDSS is derived from an identification scheme [SSH11]. Unlike other MQ-based signatures, such as Rainbow, the 5-pass version of MQDSS enjoys a formal security guarantee. The reduction in [CHR$^+$16] is based on the original rewinding technique of [PS96b]. Note that the latter was proposed for signature schemes derived from 3-pass IDS and, as such, deals with only single rewinding/forking. However, due to the use of 5-pass IDS as the underlying primitive, the security reduction of 5-pass MQDSS is significantly more involved and requires more than one rewinding. This is addressed in [CHR$^+$16] by invoking the splitting-based forking of [PS96b] multiple times. In addition, the security reduction does not follow any modular approach, like [BN06, HKL19]. Hence, deriving the success probability of the underlying forking algorithm and the associated tightness measurement for the 5-pass MQDSS appears to be quite complicated and is not provided in [CHR$^+$16]. In fact, the paper explicitly mentions that deriving exact probability bound leads only to a complicated statement ([CHR$^+$16, see the statement after Theorem 4.8]). In other words, the security proof in [CHR$^+$16] is qualitative in nature only. Interestingly, the lack of tightness in the reduction actually allows an attack [KZ20] on the concrete parameter choices of 5-pass MQDSS in NIST submission [NIS19] without invalidating the asymptotic security assurance of [CHR$^+$16].

The same 5-pass IDS scheme [SSH11] has been utilised by Petzoldt et al. [PSM17] to construct a multivariate blind signature scheme (henceforth referred to as MBSS). To the best of our knowledge, this is the only blind signature scheme available in the MQ-setting. The authors provide only a proof sketch in a significantly restricted model [PSM17], but the scheme was recently broken by Beullens [Beu24]. Recently, the 3-pass version of the IDS in [SSH11] has been used to construct an id-based signature in [CDP21]. The authors essentially follow the Bellare-Neven forking abstraction [BN06] with some suitable adaptation catering to their particular setting.

A crucial element of all the security reductions proposed in this paper is the modular application of forking. So, it is prudent to briefly recall the existing forking/rewinding abstractions. In [PS96b], Pointcheval and Stern introduced a novel proof technique called *rewinding* to argue the security of schemes, such as the Schnorr signature [Sch89], which can be obtained from an identification scheme through the Fiat-Shamir transform [FS86]. In follow-up works [PS96a, PS00], Pointcheval and Stern further consolidated and applied the rewinding strategy to the security argument of blind signature schemes, such as Okamoto-Schnorr [Oka92] and Okamoto-Guillou-Quisquater [Oka92]. Recently, Hauck et al. [HKL19] proposed a modular framework for blind signatures from any linear function family with certain properties. A core element of their modular approach is a generalisation of the rewinding technique [PS00].

Informally, in rewinding (a.k.a. *oracle replay attack*), an attacker $\mathcal{A}$ is run more than once on some related input. In all these runs of $\mathcal{A}$, the internal coin and a part of the input until a particular point of execution (called the *forking-index*) remain fixed but differ after that index. The probability of success in rewinding is captured through a fundamental lemma, called the "Splitting Lemma" in [PS96b]. Later, Bellare and Neven [BN06] further formalised this abstraction in the form of an algorithm, called the *forking algorithm*. Unlike [PS96b], the analysis of the forking in [BN06] is more algorithmic and concrete. The forking abstraction of [BN06] has been further generalised to the case of multiple forking [BPW12, CK16] with applications in proxy signature, id-based signature, zero knowledge proof, etc. [GG09, BPW12, CKK12, CMW12, BTZ22].

Here we give an informal and high-level idea of the forking algorithm from [BN06]. Let $\mathcal{B}$ be an algorithm that takes $(I, \omega, \boldsymbol{h}) \in \mathcal{I} \times \Omega \times \mathsf{R}^{\nu}$ and outputs (J, σ), where $J \in [0, \nu]$ is called the forking index, σ is called side output, I is an instance of some computational problem such as discrete log, ω is the internal random coin of $\mathcal{B}$ and $\boldsymbol{h}$ is a random vector of size ν. Let $\mathsf{Succ}(\mathcal{B})$ denote the probability of the event "$J \neq 0$", where the probability is taken over the random choice of $(I, \omega, \boldsymbol{h})$. Assume that the initial run on random input $(I, \omega, \boldsymbol{h})$ to $\mathcal{B}$ produces the output (J, σ). If $J \neq 0$, then set $\boldsymbol{h}' = (h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_\nu)$, where $h'_J, \ldots, h'_\nu$ are freshly sampled from R. Run $\mathcal{B}$ on $(I, \omega, \boldsymbol{h}')$ to obtain the output (J', σ'). If $J = J' \neq 0$ and $h_J \neq h'_J$, then we say that the forking is successful as (σ, σ') can be used to obtain a solution of the problem instance I. The associated forking lemma provides a bound on this probability of success. Notice that in the above forking, the entire input domain (which includes random coins) is split around the forking index J into two subdomains. In two different runs of $\mathcal{B}$, the part of the input coming from the first subdomain remains the same, while the part coming from the second subdomain is freshly sampled. In other words, splitting takes place at a single point and the number of rewinding/forking (i.e., the number of runs excluding the initial run) is 1. Henceforth, we would refer to this type of forking as "1-Single-Splitting-Forking". Rewinding at more than one index, on the other hand, is termed multiple forking [BCJ08, BPW12].

In their modular treatment of blind signature security, [HKL19] introduced "Subset Splitting Lemma" which can be seen as a generalisation of the original splitting lemma [PS96b]. Using it, they proved their "Subset Forking Lemma", which essentially quantifies the success probability of the underlying forking algorithm. The authors also showed that the forking algorithm [BN06] and its analysis can be derived from the subset forking lemma. [HKL19] uses a linear function family to construct blind signatures, and their modular security reductions utilise this linearity. They instantiated many existing blind signature schemes [Oka92, Oka92, PS97], where the security of the underlying linear function families rely on the DL or RSA problem.

As noted above, most existing blind signature schemes have been studied in the linear settings of DL and RSA where the constructions as well as security reductions directly or indirectly exploit the linear property of the underlying trapdoor functions. For example, consider the RSA-based blind signature, where a message can be blinded simply by multiplying with an appropriate randomizer. Such an approach does not work in non-linear MQ-setting. So, in MBSS, a message is blinded by introducing another non-linear map to the existing system, and the modular approach of [HKL19] is not applicable. As a consequence, unlike blind signatures in the setting of DL or RSA, one single underlying hard problem (like one-more RSA) appears to be insufficient to argue security of the MBSS. While we relax the 'one-more' restriction in our analogous problem, PR-mSTI (Definition 2.19), we need to consider an additional problem, CMQ (Definition 2.16). Otherwise, OMF security can be broken (see Section 1.1). Overall, the non-linear setting significantly complicates the security reduction.

Note that all the security reductions in this paper consist of three different types of algorithms: a wrapper[1], a forking algorithm and a problem solver. While designing these algorithms, we need to take into consideration the underlying non-linear setting so that the security can be based on the intractability of the corresponding hard problem(s). Note that the forking abstractions developed here suffice to produce enough transcripts to extract the desired witness. However, they are agnostic to the underlying (non-)linearity. Notwithstanding our focus on the application of these forking lemmas in a non-linear setting, there is nothing inherently non-linear in them. Hence, it does not rule out their future applications in a linear setting.

Our Result. In this work, we provide formal security reductions for the 3/5-pass MQDSS and multivariate blind signature scheme. All our security reductions follow a modular approach and require some new forking abstractions to achieve concrete security bounds. More precisely, the following are our contributions.

1. In Section 4, we revisit the formal security of 3-pass and 5-pass MQDSS through two separate reductions with concrete probability analysis. Recall that the existing security proof of 5-pass MQDSS is only qualitative, while no formal security reduction of 3-pass MQDSS is available in the literature. Based on our concrete security analysis of the 3-pass and 5-pass MQDSS, one can conclude that at any particular security level, the former performs better both in terms of computation and communication cost.

   For the security reduction of the 3-pass MQDSS, we formulate a forking abstraction "2-Single-Splitting" (Lemmas 3.4 & 3.9). While this is a straightforward extension of [BN06], for the sake of completeness, we reformulate it using the splitting lemma. For the reduction of the 5-pass MQDSS, we formulate a new forking abstraction "Special-(1,2)-Multi-Splitting" (Lemmas 3.8 & 3.13). In this case, the number of forking indices is 2 (that is, the input domain is divided into 3 subdomains). Although multiple splitting appeared in the literature earlier, this forking abstraction satisfies some additional constraints so that the main reduction algorithm can utilise 5-special soundness (Definition 2.14) to extract the underlying witness. Furthermore, unlike the existing splitting results [BN06, HKL19, BCJ08, BPW12, CK16], all our forking abstractions consider more than one forking in at least one forking index.

2. In [PSM17], the authors proposed a multivariate blind signature scheme which is claimed to achieve universal-one-more-unforgeability (UOMF) security. Note that UOMF is a much weaker model and the security of MBSS in the more acceptable standard OMF model [BNP+03] was left open in [PSM17]. We provide a concrete security reduction for OMF security of a slightly modified version of MBSS in Section 5.2. Our security argument essentially relies on two hard problems. The reduction requires two forking abstractions: 2-Single-Splitting and a new forking abstraction, "(2,2)-Multi-Splitting" (Lemmas 3.7 & 3.11), to reduce the security of the scheme to the intractability of the two underlying hard problems. We note that the last forking abstraction is designed to extract a pair of witnesses (for finding a solution to the underlying hard problem) through four rewindings/forkings.

1.1 An Outline of Our Approach

We provide an informal idea of our approach to security reductions using forking abstractions. In Section 3, we propose three abstractions to handle the cases of forking at one index (as in [BN06]) and more than

[1]An algorithm that creates the environment (see Sections 4 and 5) of digital signature or blind signature for an adversary.

one index (as in [BCJ08, BPW12]). As in previous modular approaches, the forger (attacker) is not directly rewound in any of our security reductions. Instead, a wrapper algorithm is created using the forger as a black-box, and the analysis works whenever the underlying wrapper is rewound.

First, consider the security reduction of 3 and 5 pass MQDSS. They are respectively built upon the underlying IDS having a single challenge round for the former and two challenge rounds for the latter. So, for 3-pass MQDSS, it suffices to fork the wrapper at a single point, and here we use the forking abstraction 2-single-splitting-forking (Corollary 3.10)[2]. For 5-pass MQDSS, on the other hand, one needs to rewind the wrapper at two different indices that essentially correspond to the two challenge rounds in the underlying IDS. This is handled through the special (1,2) multi-splitting lemma (Corollary 3.14). Unlike the security proof in [CHR+16], our modular forking based approach yields concrete probability analysis and thus a comparative measure of concrete security of the two versions of MQDSS.

Next comes the one-more unforgeability security of the MQ-based blind signature, MBSS. First, we give an informal idea of MBSS from [PSM17]. An unblinded signature is nothing but a proof of knowledge of the witness / secret $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \in \mathbb{F}^n \times \mathbb{F}^{\bar{n}}$ corresponding to the statement $(\mathcal{P}, \mathcal{R}, \boldsymbol{w})$. In other words, $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{w}$ where $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^{\bar{n}}$ and $\mathcal{R} : \mathbb{F}^m \to \mathbb{F}^m$ are the UOV public map and a random quadratic polynomial map, respectively, and $\boldsymbol{w} = \mathcal{H}_1(\mathrm{M})$ is the hash of the underlying message M. A user with a message M, first, creates a blinded message $\widetilde{\boldsymbol{w}} = \boldsymbol{w} - \mathcal{R}(\bar{\boldsymbol{z}})$ by uniformly sampling $\bar{\boldsymbol{z}} \in \mathbb{F}^{\bar{n}}$. The user then asks the signer for a signature on the blinded message $\widetilde{\boldsymbol{w}}$. Let $\boldsymbol{z}$ be the reply (that is, $\mathcal{P}(\boldsymbol{z}) = \widetilde{\boldsymbol{w}}$). The user then generates a proof by running r parallel instances of the IDS [SSH11] with $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ as a witness and $(\mathcal{P}, \mathcal{R}, \boldsymbol{w})$ as the statement followed by the Fiat-Shamir transform [FS86].

Recall that the one-more unforgeability security of the blind signature ensures the conservation of signature. Informally, for each unblinded signature, there must be a corresponding query to the blind signature oracle. Suppose that for two different messages M_i and M_j in MBSS, an attacker can find $\bar{\boldsymbol{z}}_i, \bar{\boldsymbol{z}}_j \in \mathbb{F}^{\bar{n}}$ such that $\mathcal{H}_1(\mathrm{M}_i) - \mathcal{R}(\bar{\boldsymbol{z}}_i) = \widetilde{\boldsymbol{w}}_i = \widetilde{\boldsymbol{w}}_j = \mathcal{H}_1(\mathrm{M}_j) - \mathcal{R}(\bar{\boldsymbol{z}}_j)$. So, a single query on the blinded message $\widetilde{\boldsymbol{w}}_i = \widetilde{\boldsymbol{w}}_j$ yields two valid message-signature pairs, which, in turn, break the OMF security. We say that the unblinded signatures σ_i and σ_j, respectively corresponding to M_i and M_j, form a **collision**.

In a linear setting like RSA, it is implicitly assumed and, in fact, relatively easy to show that one blind signature cannot be manipulated in this way to generate two valid message-signature pairs. However, the scenario turns out to be much more subtle in the non-linear MQ-setting and needs to be addressed carefully. Thus, the OMF security proof of MBSS discussed here deviates from the security treatment of blind signature schemes in the linear setting [HKL19].

To rule out the above attack in MBSS, we show that such a collision allows solving the following problem, called the CMQ problem: given a random quadratic map $\mathcal{R} : \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$ and a target $\boldsymbol{y} \in \mathbb{F}^m$, the task is to find a pair $(\bar{\boldsymbol{z}}_1, \bar{\boldsymbol{z}}_2) \in \mathbb{F}^{\bar{n}} \times \mathbb{F}^{\bar{n}}$ such that $\mathcal{R}(\bar{\boldsymbol{z}}_1) - \mathcal{R}(\bar{\boldsymbol{z}}_2) = \boldsymbol{y}$.[3] This turns out to be the most challenging part of the reduction. Let $(\mathrm{M}_1, \sigma_1), (\mathrm{M}_2, \sigma_2)$ be the corresponding message-signature pairs and J_1 and J_2 be the associated hash indices. We argue that the CMQ problem can be solved if one finds the corresponding witnesses for the statements $(\mathcal{P}, \mathcal{R}, \boldsymbol{w}_1)$ and $(\mathcal{P}, \mathcal{R}, \boldsymbol{w}_2)$ of the underlying IDS, where $\boldsymbol{w}_1 = \mathcal{H}_1(\mathrm{M}_1)$ and $\boldsymbol{w}_2 = \mathcal{H}_1(\mathrm{M}_2)$. In addition, $\mathcal{H}_1(\mathrm{M}_1)$ and $\mathcal{H}_1(\mathrm{M}_2)$ are programmed in such a way that $\mathcal{H}_1(\mathrm{M}_1) - \mathcal{H}_1(\mathrm{M}_2) = \boldsymbol{y}$, where $\boldsymbol{y}$ is the challenge part of the CMQ problem instance. This, in turn, requires rewinding the underlying wrapper at two forking indices J_1 and J_2. That is, the input domain (including random coins) has to be split around the indices, J_1 and J_2. Furthermore, our

[2]A reduction based on our earlier mentioned work [CDP21] leads to a similar result

[3]If one tries to formulate an analogous problem in the RSA setting, then it goes like this: find two distinct messages $\mathrm{M}_1, \mathrm{M}_2 \in \mathcal{M}$ and two random coins $r_1, r_2 \in \mathbb{Z}_N^*$ such that $\mathcal{H}(\mathrm{M}_1) \cdot r_1^e = \mathcal{H}(\mathrm{M}_2) \cdot r_2^e \mod N$, that is, $(r_1 \cdot r_2^{-1})^e = y \mod N$, where $y = \mathcal{H}(\mathrm{M}_2) \cdot \mathcal{H}(\mathrm{M}_1)^{-1} \mod N$ is a random value assuming $\mathcal{H}$ to be a random oracle. That is, the analogous problem in the RSA setting is to find a pair (r_1, r_2) for a given random $y \in \mathbb{Z}_N^*$ such that $(r_1 \cdot r_2^{-1})^e = y \mod N$. It is easy to see that this problem is equivalent to the RSA problem. Hence, unlike the MQ setting, this case does not need a separate treatment.

forking algorithm requires multiple forkings at each index. No existing forking abstraction can capture this situation. Therefore, we introduce a new forking abstraction, the (2,2)-multi-splitting-forking (Corollary 3.12). Finally, by combining the two witnesses, a solution of the CMQ problem instance can be found (for details, see Lemma 5.2).

Once a collision of the above type is ruled out, we show that a successful one-more forgery provides a solution to the following problem, called PR-mSTI. Given $(\mathcal{P}, \mathcal{R}, \boldsymbol{w})$ and an access to $\mathcal{P}$-inversion oracle, the task is to find $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \in \mathbb{F}^n \times \mathbb{F}^m$ such that $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{w}$, with the obvious restriction that $\widetilde{\boldsymbol{w}} = \boldsymbol{w} - \mathcal{R}(\bar{\boldsymbol{z}})$ must not be queried to the oracle. For this reduction, the forking algorithm based on 2-single-splitting-forking (Corollary 3.10) turns out to be sufficient, as we are using the 3-pass IDS as the underlying primitive.

Unlike existing blind signature security treatments, such as [PS00, HKL19], the forking algorithms involved in the MBSS security reduction require embedding the challenge from the PR-mSTI/CMQ problem instance into the random oracle response. Further, in addition to the blind signature oracle, the wrapper has to simulate two different random oracles for MBSS (and also for 5-pass MQDSS) using only a single random vector. All these require careful programming of the respective oracles, which is handled in our reduction through suitably defined encoding functions. For example, in the case of 5-pass MQDSS, we define two functions using the encodings that essentially correspond to the underlying hash functions. Encoding functions must ensure the correct distribution of the output random oracle values. Unlike the reduction of 5-pass MQDSS, the role of the wrapper in OMF security is a bit more involved as it has to handle queries to two different random oracles and one blind signature oracle using the same random vector. The proof employs different encodings to construct three functions that simulate the two hash functions and the randomizer involved in the underlying blind signature. The encodings also ensure that the PR-mSTI/CMQ problem instance is appropriately plugged into the random oracle.

Organisation. In Section 2, we provide some background on multivariate polynomials, relevant hardness assumptions, and the definition of commitment, digital signature, blind signature, and identification schemes. In Section 3, we first formulate some fundamental lemmas and then using these lemmas develop some forking abstractions. In Section 4, we provide concrete security reductions for 3-pass and 5-pass MQDSS and analyse their comparative security. In Section 5, we provide a concrete OMF security reduction of multivariate-based blind signature. Finally, we conclude in Section 6. The appendix contains some relevant definitions and omitted proofs.

2 Preliminaries

Basic notation, setting of multivariate quadratic polynomials (MQ), relevant hardness assumptions in the MQ setting, and the UOV signature scheme are defined in this section. The syntax and security definitions of the commitment scheme and the blind signature scheme are also provided in this section along with the syntax and properties of identification scheme, while some instantiations in the MQ setting are reproduced in the Appendix B.

Notations. For $a, b \in \mathbb{N} \cup \{0\}$, define $[a, b] = \{x \in \mathbb{N} \cup \{0\} : a \leq x \leq b\}$ and when $b \in \mathbb{N}$, define $[b] = [1, b]$. For a set X, we write $x \xleftarrow{\$} X$ to mean that x is drawn uniformly at random from X. For an algorithm A and its input x, the notation $y \longleftarrow A(x)$ denotes that when A is run on x, it outputs y. When A is a randomised algorithm, it takes a randomness ω as input in addition to x. If the random coin ω is supplied to A, then we write $y \longleftarrow A(x; \omega)$. For an oracle $\mathcal{O}(\cdot)$ and its input x, the notation $y \xleftarrow{Q} \mathcal{O}(x)$ indicates that when x is queried to $\mathcal{O}(\cdot)$, it returns y. Sometimes in algorithmic environments, we use the notation $x \longleftarrow y$ to mean x is assigned to y.

We use lowercase letters in boldface, for example, $\boldsymbol{h}$ to denote vector. The notation $\boldsymbol{h}_{[j]}$ means the first j entries of $\boldsymbol{h}$, i.e., $h_1, \ldots, h_j$. For $1 \leq j \leq \nu$ and $\boldsymbol{x} \in X^{j-1}$, the notation $\boldsymbol{h}' \xleftarrow{\$} X^{\nu}|\boldsymbol{x}$ means that $\boldsymbol{h}'$ is uniformly sampled from X^{ν} conditioned on $\boldsymbol{h}'_{[j-1]} = \boldsymbol{x}$. For positive integers j_1, j_2 with $j_1 \leq j_2$, the notation $\boldsymbol{h}_{[j_1,j_2]}$ denotes entries $\boldsymbol{h}$ from index j_1 to j_2, i.e., $\boldsymbol{h}_{[j_1,j_2]} = h_{j_1}, h_{j_1+1}, \ldots, h_{j_2}$.

For algorithms A and B and their respective inputs x and y, we write $z \xleftarrow{\$} \langle A(x), B(y) \rangle$ to denote that A and B interact with each other in a common interactive protocol and finally return z. The output value z could be produced by A or B alone, or together by both A and B. During their interactions, the messages exchanged between A and B constitute the transcript of the protocol and will be denoted by $\mathsf{Trans}\left(\langle A(x), B(y) \rangle\right)$.

2.1 Basic Cryptographic Primitives

Definition 2.1 (Commitment Scheme)**.** A non-interactive commitment scheme consists of three PPT algorithms: CSetup, Commit and Open.

- CSetup: It takes a security parameter κ and outputs a public commitment key ck.
- Commit: It takes as input a message $\mathrm{M} \in \mathcal{M}$, the public commitment key ck and returns a pair $(\mathsf{ct}, \mathsf{dct})$, where ct is called commitment on the message M and dct is called decommitment.
- Open: Takes a pair $(\mathsf{ct}, \mathsf{dct})$, the public commitment key ck as input, and returns M or $\perp$.

For correctness, it is required that $\mathsf{Open}(\mathsf{ck}, \mathsf{Commit}(\mathsf{ck}, \mathrm{M})) = \mathrm{M}$ for all message $\mathrm{M} \in \mathcal{M}$.

If τ is the randomness involved in Commit, then we can write $(\mathsf{ct}, \mathsf{dct}) \longleftarrow \mathsf{Commit}(\mathsf{ck}, \mathrm{M}; \tau)$. Note that in most of the constructions, τ appears as part of the decommitment dct. In all constructions considered in this paper, we do not use Open. Instead, given $(\mathsf{ct}, \mathrm{M}, \tau)$ we simply check whether $\mathsf{ct} \overset{?}{=} \mathsf{Commit}(\mathsf{ck}, \mathrm{M}; \tau)$. Throughout this paper, we neither explicitly mention ck nor the random coin τ involved in Commit.

Definition 2.2 (Hiding)**.** A commitment scheme is said to have statistical hiding property if for all (possibly quantum and computationally unbounded) algorithms $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Hiding}}(\kappa) := \left| \Pr\left[b = b'\right] - \frac{1}{2} \right|$$

in $\mathsf{Exp}_{\mathcal{A}}^{\text{Hiding}}(\kappa)$ (defined in Figure 1) is a negligible function in κ.

Remark 2.1. If $\mathsf{Adv}_{\mathcal{A}}^{\text{Hiding}}(\kappa) = 0$ (in Definition 2.2), then the hiding property is called *perfect hiding*. On the other hand, if $\mathcal{A}$ is a PPT adversary and $\mathsf{Adv}_{\mathcal{A}}^{\text{Hiding}}(\kappa)$ is negligible, then it is called *computational hiding*.

Definition 2.3 (Binding)**.** A commitment scheme is said to have computational binding property if for all (possibly quantum) PPT algorithms $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\text{Binding}}(\kappa) := \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\text{Binding}}(\kappa) = 1\right]$$

in $\mathsf{Exp}_{\mathcal{A}}^{\text{Binding}}(\kappa)$ (defined in Figure 1) is a negligible function in κ.

Definition 2.4 (Signature Scheme)**.** It consists of three PPT algorithms - KeyGen, Sign and Ver.

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\text{Hiding}}(\kappa)}$:

1. $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(1^\kappa)$
2. $(\mathrm{M}_0, \mathrm{M}_1, \mathsf{st}) \longleftarrow \mathcal{A}(\mathsf{ck})$
3. $b \xleftarrow{\$} \{0, 1\}$
4. $(\mathsf{ct}_b, \mathsf{dct}_b) \xleftarrow{\$} \mathsf{Commit}(\mathsf{ck}, \mathrm{M}_b)$
5. $b' \longleftarrow \mathcal{A}(\mathsf{ck}, \mathsf{ct}_b, \mathsf{st})$

$\underline{\mathsf{Exp}_{\mathcal{A}}^{\text{Binding}}(\kappa)}$:

1. $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(1^\kappa)$
2. $(\mathsf{ct}, \mathsf{dct}, \mathsf{dct}') \longleftarrow \mathcal{A}(\mathsf{ck})$
3. $\mathrm{M} \longleftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{ct}, \mathsf{dct})$
4. $\mathrm{M}' \longleftarrow \mathsf{Open}(\mathsf{ck}, \mathsf{ct}, \mathsf{dct}')$
5. return 1 if $\mathrm{M} \neq \mathrm{M}'$ and $\mathrm{M}, \mathrm{M} \neq \perp$, else return 0

Figure 1: Experiments for Hiding (left) and Binding (right) of commitment scheme.

- KeyGen: It takes as input a security parameter κ and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
- Sign: It takes as input a message $\mathrm{M} \in \mathcal{M}$ and a secret key sk, and outputs a signature σ, where $\mathcal{M}$ is the message space.
- Ver: It takes as input a message-signature pair (M, σ) and a public key pk. It outputs a value of 1 if (M, σ) is a valid message-signature pair, else it outputs 0.

Correctness: For all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and for all messages $\mathrm{M} \in \mathcal{M}$, it is required that

$$\mathsf{Ver}(\mathsf{pk}, \mathrm{M}, \mathsf{Sign}(\mathsf{sk}, \mathrm{M})) = 1.$$

Next, we define the security model of signature scheme. A security notion which is very useful in practice is called existentially unforgeable under chosen message attack (EUF-CMA).

Definition 2.5 (EUF-CMA)**.** A signature scheme is said to be existentially unforgeable under chosen message attack (EUF-CMA) if for all (possibly quantum) PPT algorithms $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-CMA}}(\kappa) := \Pr\left[\mathsf{Ver}(\mathsf{pk}, \mathrm{M}^*, \sigma^*) = 1 \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa);\\ (\mathrm{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sign}}}(\mathsf{pk})\end{array}\right]$$

is a negligible function in κ, where $\mathcal{A}$ is provided access to the sign oracle $\mathcal{O}_{\mathsf{sign}}$ with a natural restriction that $\mathrm{M}^* \neq \mathrm{M}$ for all messages M queried to $\mathcal{O}_{\mathsf{sign}}$. Note that $\mathcal{O}_{\mathsf{sign}}$ acts as the original signing algorithm. More precisely, it takes a query message and an underlying secret key sk (associated with the public key pk), runs the signing algorithm, and returns the output.

Remark 2.2. When $\mathcal{A}$ does not have access to $\mathcal{O}_{\mathsf{sign}}$, the security model is called existentially unforgeable under no-message attack (EUF-NMA). This model is weaker than EUF-CMA.

Definition 2.6 (Blind Signature Scheme [PSM17])**.** It consists of three PPT algorithms - KeyGen, Sign and Ver.

- KeyGen: It takes as input a security parameter κ and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.
- Sign: It is an interactive protocol between a signer S and a user U. The inputs to U and S are $(\mathsf{pk}, \mathrm{M})$ and $(\mathsf{pk}, \mathsf{sk})$ respectively. At the end of the interaction, S outputs $\mathsf{status} \in \{\mathsf{not\text{-}completed}, \mathsf{completed}\}$ and U outputs $\sigma \in \Sigma \cup \{\perp\}$, where Σ is the signature space.

  Notationally, the output of the protocol will be written as $(\sigma, \mathsf{status}) \longleftarrow \langle \mathsf{U}(\mathsf{pk}, \mathrm{M}), \mathsf{S}(\mathsf{pk}, \mathsf{sk}) \rangle$.

- Ver: It takes as input a message-signature pair (M, σ) and a public key pk. It outputs 1 if (M, σ) is a valid message-signature pair under pk, else it outputs 0.

Correctness: For all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and for all messages $\mathrm{M} \in \mathcal{M}$, it is required that if

$$(\sigma, \mathsf{status}) \longleftarrow \langle \mathsf{U}(\mathsf{pk}, \mathrm{M}), \mathsf{S}(\mathsf{pk}, \mathsf{sk}) \rangle \text{ and } \mathsf{status} = \mathsf{completed}, \text{ then } \mathsf{Ver}(\mathrm{M}, \sigma, \mathsf{pk}) = 1.$$

Remark 2.3. Typically, in the interactive signing algorithm, a user U constructs a blinded message $\widetilde{\mathrm{M}}$ from the original message M using secret ephemeral information and then sends $\widetilde{\mathrm{M}}$ to a signing authority S. The signer S signs the blinded message $\widetilde{\mathrm{M}}$ and returns the blind signature $\widetilde{\sigma}$ to the user U. The user then generates the unblinded signature σ on M from $\widetilde{\sigma}$ using the same secret ephemeral information.

A blind signature scheme needs to satisfy two security attributes: blindness and one-more unforgeability (OMF). In [PSM17], the authors considered a weaker notion of OMF, called universal one-more unforgeability (UOMF) to argue security of their multivariate blind signature scheme. The authors also established the blindness of their proposal. In this paper, our focus is on unforgeability so we skip the definition of blindness here and refer the interested readers to [PSM17] instead.

Definition 2.7 (UOMF [PSM17])**.** A blind signature scheme is said to be universal one-more unforgeable if for all (possibly quantum) PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{UOMF}}(\kappa) := \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UOMF}}(\kappa) = 1\right]$$

in $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UOMF}}(\kappa)$ (defined in Figure 2) is a negligible function in κ, where $\mathcal{A}$ is given access to blind signature oracle $\mathcal{O}_{\mathsf{BS}}$ to which it can make at most ℓ many queries.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{UOMF}}(\kappa)$:

  1. $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(1^\kappa)$
  2. $\left\{(\mathrm{M}_i, \sigma_i)_{i \in [\ell]}, \mathsf{st}\right\} \longleftarrow \mathcal{A}_1^{\mathcal{O}_{\mathsf{BS}}}(\mathsf{pk})$
  3. return 0 if $\exists i \in [\ell]$ s.t $\mathsf{Ver}(\mathrm{M}_i, \sigma_i, \mathsf{pk}) = 0$ or $\exists i, j \in [\ell]$
     s.t $\mathrm{M}_i = \mathrm{M}_j$ with $i \neq j$
  4. $\mathrm{M} \xleftarrow{\$} \mathcal{M} \setminus \{\mathrm{M}_1, \ldots, \mathrm{M}_\ell\}$
  5. $\sigma \longleftarrow \mathcal{A}_2(\mathrm{M}, \mathsf{st})$
  6. return 1 if $\mathsf{Ver}(\mathrm{M}, \sigma, \mathsf{pk}) = 1$, else return 0
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa)$:

  1. $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(1^\kappa)$
  2. $\left\{(\mathrm{M}_i, \sigma_i)_{i \in [\ell+1]}\right\} \longleftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{BS}}}(\mathsf{pk})$
  3. return 1 if $\forall i \in [\ell+1]$, $\mathsf{Ver}(\mathrm{M}_i, \sigma_i, \mathsf{pk}) = 1$ and $\forall i, j \in$
     $[\ell+1]$ with $i \neq j$, $\mathrm{M}_i \neq \mathrm{M}_j$
  4. return 0
```

Figure 2: Experiments for UOMF (left) and OMF (right) of blind signature scheme.

Remark 2.4. In both experiments, UOMF and OMF, the blind signature oracle $\mathcal{O}_{\mathsf{BS}}$ models the interactive stage of a blind signature scheme, in which the oracle takes the underlying secret key sk and a blinded message $\widetilde{\mathrm{M}}$ as input, and outputs a blind signature $\widetilde{\sigma}$. Note that in the first phase of the UOMF experiment, $\mathcal{A}_1$ outputs ℓ message-signature pairs and a state st. In the second phase, $\mathcal{A}_2$ receives the internal state st and a random message M (distinct from the messages output by $\mathcal{A}_1$), and produces a forgery on M.

Definition 2.8 (OMF)**.** A blind signature scheme is said to be one-more unforgeable if for all (possibly quantum) PPT algorithms $\mathcal{A}$, the advantage

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa) := \Pr\left[\mathsf{Exp}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa) = 1\right]$$

in $\mathsf{Exp}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa)$ (defined in Figure 2) is a negligible function in κ, where $\mathcal{A}$ is provided access to blind signature oracle $\mathcal{O}_{\mathsf{BS}}$ to which it can make at most ℓ many queries.

For the definition of identification scheme (IDS) and related properties, we closely follow [CHR+16].

Definition 2.9 (IDS)**.** An identification scheme consists of three PPT algorithms - KeyGen, a prover algorithm P and a verifier algorithm V.

- KeyGen: It takes as input a security parameter κ and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{P}(\mathsf{pk}, \mathsf{sk})$ and $\mathsf{V}(\mathsf{pk})$ are the interactive algorithms involved in a common protocol, where the prover P speaks first. At the end of the interaction, the verifier V outputs a bit b, where $b = 1$ and $b = 0$ indicate *accept* and *reject*, respectively.

Correctness: For all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$, it is required that $\Pr[b = 1 : \ b \longleftarrow \langle \mathsf{P}(\mathsf{pk}, \mathsf{sk}), \mathsf{V}(\mathsf{pk})\rangle] = 1$.

Remark 2.5. We often refer to sk and pk as witness and statement respectively. Note that for a statement, there could be more than one witness.

Definition 2.10 (Soundness with knowledge error μ)**.** Let $\mu \in \mathbb{N}$. An identification scheme $\mathsf{IDS} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ is said to be sound with knowledge error μ if for all PPT $\mathcal{A}$, we have

$$\Pr[b = 1 : \ (\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(\kappa); \ b \longleftarrow \langle \mathcal{A}(\mathsf{pk}, \perp), \mathsf{V}(\mathsf{pk})\rangle] \leq \mu + \mathsf{negl}(\kappa).$$

Definition 2.11 (Statistical HVZK)**.** An identification scheme $\mathsf{IDS} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ is said to be statistical honest verifier zero-knowledge (HVZK) if there exists a simulator Simu such that the statistical distance between the following ensembles is negligible in κ:

1. $\{(\mathsf{pk}, \mathsf{sk}, \pi) : \ (\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(\kappa); \ \pi \longleftarrow \mathsf{Trans}(\langle \mathsf{P}(\mathsf{pk}, \mathsf{sk}), \mathsf{V}(\mathsf{pk})\rangle)\}$,
2. $\{(\mathsf{pk}, \mathsf{sk}, \pi) : \ (\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KeyGen}(\kappa); \ \pi \longleftarrow \mathsf{Simu}(\mathsf{pk})\}$.

Next, we define the canonical $(2n + 1)$-pass IDS as follows.

Definition 2.12 (Canonical $(2n + 1)$-Pass IDS)**.** Let $\mathsf{IDS} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ be an identification scheme with n challenge spaces $\mathsf{ChS}_1, \ldots, \mathsf{ChS}_n$. Then the IDS is said to be canonical $(2n + 1)$-pass identification scheme if the prover and the verifier can be split into $(n + 1)$ subroutines $\mathsf{P} = (\mathsf{P}_0, \mathsf{P}_1, \ldots, \mathsf{P}_n)$ and $\mathsf{V} = (\mathsf{V}_1, \ldots, \mathsf{V}_n, \mathsf{V}_{n+1})$ respectively such that

- $\mathsf{P}_0(\mathsf{sk})$ computes the initial commitment ct and sends to V as the first message.
- V_i for $1 \leq i \leq n$ samples i-th challenge $\mathsf{ch}_i \xleftarrow{\$} \mathsf{ChS}_i$ after receiving $(i - 1)$-th response rs_{i-1} and sends ch_i to P. Note that when $i = 1$, $\mathsf{rs}_{i-1} = \mathsf{ct}$.
- $\mathsf{P}_i(\mathsf{sk}, \mathsf{ct}, \mathsf{ch}_1, \mathsf{rs}_1, \mathsf{ch}_2, \ldots, \mathsf{rs}_{i-1}, \mathsf{ch}_i)$ for $1 \leq i \leq n$ computes i-th response rs_i and sends to V.
- Finally, V_{n+1} decides to accept or reject based on pk and π, where $\pi = (\mathsf{ct}, \mathsf{ch}_1, \mathsf{rs}_1, \ldots, \mathsf{ch}_n, \mathsf{rs}_n)$ is called the transcript.

Remark 2.6. In general for practical use of IDS, we want the knowledge error μ to be $\mathsf{negl}(\kappa)$. If μ is non-negligible in κ, then [AF22] has shown that by running IDS r-times in parallel, one can make the knowledge error $\mu^r = \mathsf{negl}(\kappa)$ by considering r sufficiently large. For simplicity of understanding, we use the notation

$\mathsf{IDS}^r = (\mathsf{KeyGen}, \mathsf{P}^r, \mathsf{V}^r)$ for r parallel instances of $(2n+1)$-pass IDS, where $\mathsf{P}^r = (\mathsf{P}_0^r, \mathsf{P}_1^r, \ldots, \mathsf{P}_n^r)$ and $\mathsf{V}^r = (\mathsf{V}_1^r, \ldots, \mathsf{V}_n^r, \mathsf{V}_{n+1}^r)$. Furthermore, a transcript of IDS^r is of the form $\pi = (\mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \ldots, \mathbf{ch}_n, \mathbf{rs}_n)$, where $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_r)$, and for $1 \le i \le n$, $\mathbf{ch}_i = (\mathsf{ch}_{i,1}, \ldots, \mathsf{ch}_{i,r})$ and $\mathbf{rs}_i = (\mathsf{rs}_{i,1}, \ldots, \mathsf{rs}_{i,r})$. Hence, π is nothing but the collection of r many transcripts of IDS, i.e., $\pi = (\pi_1, \ldots, \pi_r)$ with $\pi_i = (\mathsf{ct}_i, \mathsf{ch}_{1,i}, \mathsf{rs}_{1,i}, \ldots, \mathsf{ch}_{n,i}, \mathsf{rs}_{n,i})$ for $1 \le i \le r$. Note that $\mathsf{V}^r(\mathsf{pk}, \pi) = 1$ if $\mathsf{V}_{n+1}^i(\mathsf{pk}, \pi_i) = 1$ for $\forall i \in [r]$, else 0.

When $n = 1$ and $n = 2$, then we have canonical 3-pass IDS and 5-pass IDS, and the respective transcripts can be written as $\pi = (\mathsf{ct}, \mathsf{ch}, \mathsf{rs})$ and $\pi = (\mathsf{ct}, \mathsf{ch}_1, \mathsf{rs}_1, \mathsf{ch}_2, \mathsf{rs}_2)$. In the following, we define the notion of special soundness for 3-pass IDS and 5-pass IDS.

Definition 2.13 (3-Special Soundness). A 3-pass identification scheme $\mathsf{IDS} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ with $|\mathsf{ChS}| \ge 3$ is said to satisfy 3-special soundness if there exists an extractor Extr such that given any three accepted transcripts of the form $\pi = (\mathsf{ct}, \mathsf{ch}, \mathsf{rs})$, $\pi' = (\mathsf{ct}, \mathsf{ch}', \mathsf{rs}')$ and $\pi'' = (\mathsf{ct}, \mathsf{ch}'', \mathsf{rs}'')$ with $\mathsf{ch} \ne \mathsf{ch}' \ne \mathsf{ch}'' \ne \mathsf{ch}$, Extr can efficiently compute a witness sk.

Definition 2.14 (5-Special Soundness). A 5-pass identification scheme $\mathsf{IDS} = (\mathsf{KeyGen}, \mathsf{P}, \mathsf{V})$ with $|\mathsf{ChS}_1| \ge 2$ and $|\mathsf{ChS}_2| \ge 2$ is said to satisfy 5-special soundness if there exists an extractor Extr such that given any four accepted transcripts of the form $\pi = (\mathsf{ct}, \mathsf{ch}_1, \mathsf{rs}_1, \mathsf{ch}_2, \mathsf{rs}_2)$, $\pi' = (\mathsf{ct}, \mathsf{ch}_1', \mathsf{rs}_1', \mathsf{ch}_2', \mathsf{rs}_2')$, $\pi'' = (\mathsf{ct}, \mathsf{ch}_1'', \mathsf{rs}_1'', \mathsf{ch}_2'', \mathsf{rs}_2'')$, and $\pi''' = (\mathsf{ct}, \mathsf{ch}_1''', \mathsf{rs}_1''', \mathsf{ch}_2''', \mathsf{rs}_2''')$ with $\mathsf{ch}_1 = \mathsf{ch}_1' \ne \mathsf{ch}_1'' = \mathsf{ch}_1'''$ and $\mathsf{ch}_2 = \mathsf{ch}_2'' \ne \mathsf{ch}_2' = \mathsf{ch}_2'''$, Extr can efficiently compute a witness sk.

Note that it is not a straightforward extension of 3-special soundness; instead, the restrictions on the challenges are bit complicated. Due to this complication, the security proof of the signature scheme based on 5-pass IDS, like MQDSS, is more involved than its 3-pass version (see Section 4). In [CHR$^+$16], this soundness is referred to as the $q2$-extractor, when $|\mathsf{ChS}_1| = q$ and $|\mathsf{ChS}_2| = 2$.

2.2 MQ Setting and Hardness Assumptions

Suppose v, o and n are three positive integers such that $n = v + o$, where v represents the number of vinegar variables, o represents the number of oil variables and n denotes the total number of variables. Let m denote the number of equations in a system. Without loss of generality, we assume that of the n variables, the first v variables are vinegar variables, and the rest o variables are oil variables. If $\boldsymbol{x} = (x_1, \ldots, x_v, x_{v+1}, \ldots, x_{v+o})$, then we write $\boldsymbol{x} = (\boldsymbol{x}_v, \boldsymbol{x}_o)$, where $\boldsymbol{x}_v = (x_1, \ldots, x_v)$ and $\boldsymbol{x}_o = (x_{v+1}, \ldots, x_{v+o})$. In this setting, the number of equations is considered the same as the number of oil variables, i.e., $m = o$.

Let $\mathbb{F}$ be a fixed finite field. By a quadratic polynomial map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ of oil-vinegar type [Pat97], we mean $\mathcal{F} = (f^{(1)}, \ldots, f^{(m)})$, where each $f^{(k)} : \mathbb{F}^n \to \mathbb{F}$ is a quadratic polynomial of oil-vinegar type, i.e., of the form:

$$f^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{v} \sum_{j=i}^{n} \alpha_{ij}^{(k)} \cdot x_i x_j + \sum_{i=1}^{n} \beta_i^{(k)} \cdot x_i + \gamma^{(k)} \tag{1}$$

where $\alpha_{ij}^{(k)}, \beta_i^{(k)}, \gamma^{(k)} \in \mathbb{F}$ for $k \in [m]$. When $v \approx 2m$, then the above map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ is called the UOV central map [KPG99]. Note that when the vinegar variables are fixed, the system $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ becomes linear and can be solved efficiently.

Affine Maps. By invertible affine map, we mean $\mathcal{T} = (A, \boldsymbol{a}) \in \mathsf{GL}_n(\mathbb{F}) \times \mathbb{F}^n$, where $\mathsf{GL}_n(\mathbb{F})$ denotes the set of all $n \times n$ invertible (i.e., non-singular) matrices over $\mathbb{F}$. For $\boldsymbol{x} \in \mathbb{F}^n$, define $\mathcal{T}(\boldsymbol{x}) := A\boldsymbol{x} + \boldsymbol{a}$. So, $\mathcal{T}$ is a map from $\mathbb{F}^n$ onto $\mathbb{F}^n$. The notation $\mathsf{invAff}(\mathbb{F}^n, \mathbb{F}^n)$ denotes the set of all affine invertible maps from $\mathbb{F}^n$ onto $\mathbb{F}^n$. Similarly, we can define $\mathsf{invAff}(\mathbb{F}^m, \mathbb{F}^m)$.

UOV Signature Scheme. In the following, we briefly reproduce the three main algorithms of the UOV signature scheme [KPG99].

- $\mathsf{UOV.KeyGen}(\kappa)$. Let (v, o, q) denote the parameters of the UOV signature defined by the security parameter κ. Set $n = v + o$ and $m = o$. Choose a finite field $\mathbb{F}$ of cardinality q. Let $\mathcal{H} : \mathcal{M} \to \mathbb{F}^m$ be a cryptographic hash function. Randomly choose a UOV central map $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ and an affine map $\mathcal{T} \in \mathsf{invAff}(\mathbb{F}^n, \mathbb{F}^n)$, and then set the UOV public map and secret map as $\mathcal{P} = \mathcal{F} \circ \mathcal{T} : \mathbb{F}^n \to \mathbb{F}^m$ and $(\mathcal{F}, \mathcal{T})$, respectively. Return $\mathsf{pk} = (\mathcal{P}, \mathcal{H})$ and $\mathsf{sk} = (\mathcal{F}, \mathcal{T}, \mathcal{H})$.
- $\mathsf{UOV.Sign}(\mathsf{sk}, \mathrm{M})$. Find an $\boldsymbol{x} \in \mathbb{F}^n$ such that $\mathcal{P}(\boldsymbol{x}) = \boldsymbol{y}$, where $\boldsymbol{y} = \mathcal{H}(\mathrm{M})$. This is done by first fixing the vinegar variables, followed by solving the reduced system using the oil variables. In fact, choose a random assignment $\boldsymbol{x}'_v$ for vinegar variables $\boldsymbol{x}_v$, then solve the linear system $\mathcal{F}(\boldsymbol{x}'_v, \boldsymbol{x}_o) = \boldsymbol{y}$ in oil variables $\boldsymbol{x}_o$. If a solution does not exist, consider a different assignment for the vinegar variables. Let $\boldsymbol{x}'_o$ be a solution of the reduced system. Set $\sigma = \mathcal{T}^{-1}(\boldsymbol{x})$, where $\boldsymbol{x} = (\boldsymbol{x}'_v, \boldsymbol{x}'_o)$ and return the signature σ.
- $\mathsf{UOV.Ver}(\mathsf{pk}, \mathrm{M}, \sigma)$. The signature is accepted if and only if $\mathcal{P}(\sigma) = \mathcal{H}(\mathrm{M})$.

Remark 2.7. Sometimes, we use the notation $\mathsf{UOVInv}(\mathsf{sk}^*, \boldsymbol{y})$ or $\mathcal{P}^{-1}(\mathsf{sk}^*, \boldsymbol{y})$ for $\mathsf{UOV.Sign}$ for the target value $\boldsymbol{y} \in \mathbb{F}^m$, where $\mathsf{sk}^* = (\mathcal{F}, \mathcal{T})$.

Polynomial Classes. Let $\mathcal{P}(\mathbb{F}^n, \mathbb{F}^m)$ denote the set of all quadratic polynomial maps $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$. Let $\mathcal{F}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m)$ be the set of all UOV central maps $\mathcal{F} : \mathbb{F}^n \to \mathbb{F}^m$ as defined above. Define $\mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) = \{\mathcal{F} \circ \mathcal{T} : \ \mathcal{F} \in \mathcal{F}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \text{ and } \mathcal{T} \in \mathsf{invAff}(\mathbb{F}^n, \mathbb{F}^n)\}$. This is essentially the set of all UOV public maps. It is obvious that $\mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m)$ is a subset of $\mathcal{P}(\mathbb{F}^n, \mathbb{F}^m)$.

Polar Form. For a map $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$, its polar form $\mathsf{G} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^m$ is defined by

$$\mathsf{G}(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{P}(\boldsymbol{x} + \boldsymbol{y}) - \mathcal{P}(\boldsymbol{x}) - \mathcal{P}(\boldsymbol{y}) + \mathcal{P}(\boldsymbol{0}) = (p_i(\boldsymbol{x} + \boldsymbol{y}) - p_i(\boldsymbol{x}) - p_i(\boldsymbol{y}) + p_i(\boldsymbol{0}))_{i \in [m]}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}^n$ and $\mathcal{P} = (p_1, \ldots, p_m)$. It is easy to show that it is a bilinear map in each variable.

For polynomial maps $\mathcal{P} : \mathbb{F}^n \to \mathbb{F}^m$ and $\mathcal{R} : \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$, where $\bar{n} \in \mathbb{N}$, we treat $\Gamma = (\mathcal{P}, \mathcal{R}) : \mathbb{F}^n \times \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$ as a polynomial map defined as follows:

$$\Gamma(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = (p_i(\boldsymbol{z}) + h_i(\bar{\boldsymbol{z}}))_{i \in [m]}$$

where $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \in \mathbb{F}^n \times \mathbb{F}^{\bar{n}}$, $\mathcal{P} = (p_1, \ldots, p_m)$ and $\mathcal{R} = (h_1, \ldots, h_m)$.

Hardness Assumptions. The multivariate quadratic (MQ)-problem is one of the central problems in multivariate public-key cryptography, and its decision version is known to be NP-hard [KPG99]. Note that the security parameter κ defines several sets of parameters consisting of the number of variables, the number of equations, and the size of the underlying field. Let (n, m, q) be one such set of parameters with $m = o$ and $n = v + o$. For simplicity of the security reduction, we assume that the security parameter κ uniquely defines the corresponding parameter set. For ease of exposition, we may drop κ from the following hardness assumptions.

Definition 2.15 (MQ-problem [SSH11])**.** Given $(\mathcal{P}, \boldsymbol{y}^*) \in \mathcal{P}(\mathbb{F}^n, \mathbb{F}^m) \times \mathbb{F}^m$, find an $\boldsymbol{x}^* \in \mathbb{F}^n$ such that $\boldsymbol{y}^* = \mathcal{P}(\boldsymbol{x}^*)$. The advantage of an algorithm $\mathcal{A}$ in breaking the MQ-problem is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{MQ}}(\kappa) = \mathsf{Pr}\left[\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^* : \ (\mathcal{P}, \boldsymbol{y}^*) \xleftarrow{\$} \mathcal{P}(\mathbb{F}^n, \mathbb{F}^m) \times \mathbb{F}^m; \ \boldsymbol{x}^* \leftarrow \mathcal{A}(\mathcal{P}, \boldsymbol{y}^*)\right].$$

We say the MQ-problem is intractable if for every (possibly quantum) probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{MQ}}(\kappa)$ is a negligible function in κ.

Now, we define a new problem in the MQ-setting which we call the *conjugate MQ* (CMQ)-problem.

Definition 2.16 (CMQ-problem)**.** Given $(\mathcal{R}, \boldsymbol{y}^*) \in \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m$, find $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \in \mathbb{F}^{\bar{n}} \times \mathbb{F}^{\bar{n}}$ such that $\mathcal{R}(\boldsymbol{x}_1^*) - \mathcal{R}(\boldsymbol{x}_2^*) = \boldsymbol{y}^*$. The advantage of an algorithm $\mathcal{A}$ in breaking the CMQ-problem is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CMQ}}(\kappa) = \mathsf{Pr}\left[\mathcal{R}(\boldsymbol{x}_1^*) - \mathcal{R}(\boldsymbol{x}_2^*) = \boldsymbol{y}^* : \ (\mathcal{R}, \boldsymbol{y}^*) \xleftarrow{\$} \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m; \ (\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \leftarrow \mathcal{A}(\mathcal{R}, \boldsymbol{y}^*)\right].$$

We say the CMQ-problem is intractable if for every (possibly quantum) PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CMQ}}(\kappa)$ is a negligible function in κ.

Next, we define a problem in the MQ-setting that is no harder than the MQ-problem.

Definition 2.17 (UOV-Inversion (UOVI) problem)**.** Given $(\mathcal{P}, \boldsymbol{y}^*) \in \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathbb{F}^m$, find an $\boldsymbol{x}^* \in \mathbb{F}^n$ such that $\boldsymbol{y}^* = \mathcal{P}(\boldsymbol{x}^*)$. The advantage of an algorithm $\mathcal{A}$ in breaking the UOVI-problem is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{UOVI}}(\kappa) = \mathsf{Pr}\left[\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^* : \ (\mathcal{P}, \boldsymbol{y}^*) \xleftarrow{\$} \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathbb{F}^m; \ \boldsymbol{x}^* \leftarrow \mathcal{A}(\mathcal{P}, \boldsymbol{y}^*)\right].$$

We say the UOVI-problem is intractable if for every (possibly quantum) PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{UOVI}}(\kappa)$ is a negligible function in κ.

We now define another problem in the MQ-setting, which we call the PR-STI problem. A similar kind of problem was used in [PSM17] for arguing universal one-more unforgeability of their blind-signature scheme.

Definition 2.18 (Single Target PR Inversion (PR-STI) Problem)**.** Given $(\mathcal{P}, \mathcal{R}, \boldsymbol{y}^*) \in \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m$, find a pair $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \in \mathbb{F}^n \times \mathbb{F}^{\bar{n}}$ such that $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{y}^*$. The advantage of an algorithm $\mathcal{A}$ in breaking the PR-STI problem is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\text{PR-STI}}(\kappa) = \mathsf{Pr}\left[\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{y}^* : \ (\mathcal{P}, \mathcal{R}, \boldsymbol{y}^*) \xleftarrow{\$} \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m; \ (\boldsymbol{z}, \bar{\boldsymbol{z}}) \leftarrow \mathcal{A}(\mathcal{P}, \mathcal{R}, \boldsymbol{y}^*)\right].$$

We say the PR-STI problem is intractable if for every (possibly quantum) PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{PR-STI}}(\kappa)$ is a negligible function in κ.

The above problem is no harder[4] than the UOV inversion problem. Note that the question here is essentially to invert the one-way function $(\mathcal{P}, \mathcal{R}) : \mathbb{F}^n \times \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$ for a given target point $\boldsymbol{y} \in \mathbb{F}^m$. Its analogous version in the RSA setting is the RSA inversion problem [Cor00]. In the next, we define a variant of the PR-STI problem by providing a polynomial number of queries to the UOV-inversion oracle, but with a restriction on the queries to rule out trivial attack. Formally, the problem is defined as follows.

Definition 2.19 (Modified Single Target Inversion (PR-mSTI) Problem)**.** Given $(\mathcal{P}, \mathcal{R}, \boldsymbol{w}^*) \in \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m$ and a helper oracle $\mathcal{O}$ to calculate the inverse of $\mathcal{P}$ (i.e., $\mathcal{P}^{-1}(\mathsf{sk}^*, .)$ as mentioned in Remark 2.7), find a pair $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \in \mathbb{F}^n \times \mathbb{F}^{\bar{n}}$ such that

1. $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{w}^*$ and
2. $\widetilde{\boldsymbol{w}}$ was never queried to the helper oracle $\mathcal{O}$, where $\widetilde{\boldsymbol{w}} = \boldsymbol{w}^* - \mathcal{R}(\bar{\boldsymbol{z}})$.

Let us denote the above two conditions by cond. The advantage of an algorithm $\mathcal{A}$ in breaking the PR-mSTI problem is defined by

$$\mathsf{Adv}_{\mathcal{A}}^{\text{PR-mSTI}}(\kappa) = \mathsf{Pr}\left[\mathsf{cond} = \mathsf{true} : \ (\mathcal{P}, \mathcal{R}, \boldsymbol{w}^*) \xleftarrow{\$} \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m) \times \mathbb{F}^m; \ (\boldsymbol{z}, \bar{\boldsymbol{z}}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathcal{P}, \mathcal{R}, \boldsymbol{w}^*)\right].$$

We say the PR-mSTI problem is intractable if for every (possibly quantum) PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}_{\mathcal{A}}^{\text{PR-mSTI}}(\kappa)$ is a negligible function in κ.

[4]The two problems are equivalent when the random system $\mathcal{R}$ in PR-STI problem is treated as random oracle.

When an analogous version of this problem is considered in the setting of RSA or DL, then it can be trivially broken by manipulating the query arguments of the helper oracle. For example, in RSA-setting the problem can be trivially solved. Suppose an attacker $\mathcal{A}$ is given N (RSA modulus), $e \in \mathbb{Z}^*_{\varphi(N)}$ (public key), $y \in \mathbb{Z}_N$ (target) and access to RSA-inversion oracle. Then $\mathcal{A}$ picks $a \in \mathbb{Z}^*_N$ and makes a query $y \cdot a^e$ to the RSA-inversion oracle and gets the reply z (say). The attacker then returns $x = z \cdot a^{-1} \mod N$ as the solution to the problem. For this reason, the one-more-RSA-inversion problem is considered to argue the OMF security of the RSA based blind signature [BNP+03]. If we look closely, this attack exists mainly due to the linear structure of the RSA function. However, the PR-mSTI problem is defined in the MQ-setting which is non-linear by its nature. So, the aforementioned attack does not work for the PR-mSTI problem. In fact, the non-linearity is the key property for the security of multivariate-based cryptosystems.

Let us fix the underlying field and the parameter set of the PR-mSTI problem to be $\mathbb{F}$ and (n, m, q) with $q = |\mathbb{F}|$, $m = o$ and $n = v + o$. This means that all instances of the problem will have the same parameter set and underlying field as above. Let the collection of all instances of the problem be denoted by the set $\mathsf{Prmsti} = \mathsf{Prmsti}_1 \times \mathsf{Prmsti}_2$, where $\mathsf{Prmsti}_1 = \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m)$ and $\mathsf{Prmsti}_2 = \mathbb{F}^m$. That is, a random instance of the problem can be sampled as follows: $((\mathcal{P}, \mathcal{R}), \boldsymbol{w}^*) \xleftarrow{\$} \mathsf{Prmsti}_1 \times \mathsf{Prmsti}_2$. Note that access to the helper oracle $\mathcal{O}(\cdot)$ is also included as part of the instance. As mentioned earlier, this oracle is implemented in a manner similar to the plain signing of the UOV, that is, using the secret key $\mathsf{sk}^* = (\mathcal{F}, \mathcal{T})$ associated with the UOV public key $\mathcal{P}$.

3 Splitting Lemmas and Forking Abstractions

First we formulate some fundamental lemmas (Lemmas 3.4, 3.7 and 3.8) by extending the basic "Splitting Lemma" of [PS00, HKL19]. These fundamental lemmas will be used to develop our forking abstractions in Section 3.3. We begin by reproducing the basic splitting lemma in its original form from [PS00], stated as Lemma 3.1. For the proof, we refer to the same paper. Note that item (ii) in Lemma 3.1 follows from the definition of B_α given in equation 2. Our Proposition 3.5 (and Proposition C.1 in Appendix C) is nothing but a suitable adaptation of Lemma 3.1 and thus retains the same structure.

Lemma 3.1 (Splitting Lemma [PS00])**.** *Let X and Y be finite sets. Let $B \subset X \times Y$ be such that*

$$\Pr_{(x,y) \xleftarrow{\$} X \times Y} [(x, y) \in B] = \varepsilon.$$

For any $\alpha \leq \varepsilon$, define

$$B_\alpha = \left\{ (x, y) \in X \times Y : \Pr_{y' \xleftarrow{\$} Y} \left[(x, y') \in B \right] \geq \varepsilon - \alpha \right\}. \tag{2}$$

Then, the following statements hold for any $\alpha \leq \varepsilon$:

(i) $\Pr_{(x,y) \xleftarrow{\$} X \times Y} [(x, y) \in B_\alpha] \geq \alpha$

(ii) $\forall (x, y) \in B_\alpha$: $\Pr_{y' \xleftarrow{\$} Y} [(x, y') \in B] \geq \varepsilon - \alpha$

(iii) $\Pr_{(x,y) \xleftarrow{\$} X \times Y} [(x, y) \in B_\alpha \mid (x, y) \in B] \geq \alpha / \varepsilon.$

Claim 3.2. *Let X, Y, B and B_α be as in Lemma 3.1. Then, the following inequality holds.*

$$\Pr_{(x,y,y') \xleftarrow{\$} X\times Y\times Y}\left[(x,y) \in B_\alpha \mid (x,y) \in B \land (x,y') \in B\right] \geq \Pr_{(x,y) \xleftarrow{\$} X\times Y}\left[(x,y) \in B_\alpha \mid (x,y) \in B\right].$$

Proof. By the definition of B_α given in equation (2), the probability of a random point $(x, y) \in X \times Y$ belonging to the set B_α increases if for the same x, it is already known that there exists another point $(x, y') \in B$ for some $y' \in Y$. Therefore, for a random choice of $(x, y, y') \in X \times Y \times Y$, we can write

$$\Pr\left[(x,y) \in B_\alpha \mid (x,y) \in B \land (x,y') \in B\right] \geq \Pr\left[(x,y) \in B_\alpha \mid (x,y) \in B\right].$$

□

3.1 Single-Splitting Lemmas

We recall a formalisation of the basic splitting lemma, called "Subset Splitting Lemma" [HKL19]. For the proof, we refer to [HKL19].

Lemma 3.3 (Subset Splitting Lemma [HKL19])**.** *Let X and Y be finite sets. Let $B \subset X \times Y$ be such that*

$$\Pr_{(x,y) \xleftarrow{\$} X\times Y}\left[(x,y) \in B\right] = \varepsilon.$$

Then, for any $\alpha \leq \varepsilon$, we have

$$\Pr_{\substack{(x,y) \xleftarrow{\$} X\times Y \\ y' \xleftarrow{\$} Y}}\left[(x,y) \in B \land (x,y') \in B\right] \geq (\varepsilon - \alpha) \cdot \alpha.$$

The lemma essentially represents an algorithmic view of the splitting lemma. Here B is the subset that corresponds to the success of the event (i.e., $(x, y) \in B$) of the underlying algorithm with the probability of success being at least ε. The lemma says that if we run the algorithm twice with the input $(x, y) \in X \times Y$ and $(x, y') \in X \times Y$, respectively, then the probability of success (i.e., $(x, y) \in B \land (x, y') \in B$) after two subsequent runs of the algorithm is at least $(\varepsilon - \alpha) \cdot \alpha$ for any $\alpha \leq \varepsilon$, where $x' \xleftarrow{\$} X$ and $y', y \xleftarrow{\$} Y$. Henceforth, we shall refer to the lemma 3.3 as the "1-Single-Splitting Lemma", because splitting takes place only at a single position of the domain, resulting in two subdomains X and Y, and after the first run, a value of Y is sampled only once.

We, now extend the previous form of splitting lemma to "2-Single-Splitting Lemma", where one may think that the underlying algorithm is run for a third time but with a different random sample $y'' \in Y$. This is formally stated as follows.

Lemma 3.4 (2-Single-Splitting Lemma)**.** *Let X and Y be finite sets. Let $B \subset X \times Y$ be such that*

$$\Pr_{(x,y) \xleftarrow{\$} X\times Y}\left[(x,y) \in B\right] = \varepsilon.$$

Then, for any $\alpha \leq \varepsilon$, we have

$$\Pr_{\substack{(x,y) \xleftarrow{\$} X\times Y \\ y' \xleftarrow{\$} Y \\ y'' \xleftarrow{\$} Y}}\left[(x,y) \in B \land (x,y') \in B \land (x,y'') \in B\right] \geq (\varepsilon - \alpha)^2 \cdot \alpha^2/\varepsilon.$$

Proof. First, we look at the following conditional probability:

$$\begin{aligned}\Delta_1 &= \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y\\ y''\xleftarrow{\$}Y}}\big[(x,y'')\in B \mid (x,y)\in B\wedge(x,y')\in B\big]\\ &\geq \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y\\ y''\xleftarrow{\$}Y}}\big[(x,y'')\in B\wedge(x,y)\in B_\alpha \mid (x,y)\in B\wedge(x,y')\in B\big]\\ &= \Delta_{11}\cdot\Delta_{12},\end{aligned}$$

where

$$\begin{aligned}\Delta_{11} &= \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y\\ y''\xleftarrow{\$}Y}}\big[(x,y'')\in B \mid (x,y)\in B_\alpha\wedge(x,y)\in B\wedge(x,y')\in B\big]\\ &\geq \varepsilon-\alpha \qquad \text{[follows from item (ii) of Lemma 3.1]}\end{aligned}$$

and

$$\begin{aligned}\Delta_{12} &= \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y}}\big[(x,y)\in B_\alpha \mid (x,y)\in B\wedge(x,y')\in B\big]\\ &\geq \Pr_{(x,y)\xleftarrow{\$}X\times Y}\big[(x,y)\in B_\alpha \mid (x,y)\in B\big] \qquad \text{[due to Claim 3.2]}\\ &\geq \alpha/\varepsilon \qquad \text{[follows from item (iii) of Lemma 3.1].}\end{aligned}$$

So, we can write

$$\Delta_1 \geq (\varepsilon-\alpha)\cdot\alpha/\varepsilon. \tag{3}$$

Finally, we have

$$\begin{aligned}\Delta &= \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y\\ y''\xleftarrow{\$}Y}}\big[(x,y)\in B\wedge(x,y')\in B\wedge(x,y'')\in B\big]\\ &= \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y\\ y''\xleftarrow{\$}Y}}\big[(x,y'')\in B \mid (x,y)\in B\wedge(x,y')\in B\big]\cdot \Pr_{\substack{(x,y)\xleftarrow{\$}X\times Y\\ y'\xleftarrow{\$}Y}}\big[(x,y)\in B\wedge(x,y')\in B\big]\\ &\geq (\varepsilon-\alpha)\cdot\alpha/\varepsilon\cdot(\varepsilon-\alpha)\cdot\alpha \qquad \text{[using equation (3) and Lemma 3.3]}\\ &= (\varepsilon-\alpha)^2\cdot\alpha^2/\varepsilon.\end{aligned}$$

□

3.2 Multi-Splitting Lemmas

In this section, we consider a further generalisation of the single splitting lemmas to include splitting in multiple positions. We, therefore, refer to them as “Multi-Splitting Lemmas”. More precisely, we consider

$X \times Y \times Z$ to be our sample space. Unlike single splitting lemmas, we either fix x or (x, y) for all the runs. We therefore split the domain into either $(X, Y \times Z)$ or $(X \times Y, Z)$. The former (resp. latter) splitting is referred to as 'left' (resp. 'right') splitting. We also consider a special splitting (see Lemma 3.8), where the domain $X \times Y \times Z \times U$ is split as (X, Y, Z, U).

We now extend Lemma 3.1 to the multi-splitting environment. The following proposition is basically an extension of the basic splitting lemma.

Proposition 3.5. *Let X, Y and Z be finite sets. Let $B \subset X \times Y \times Z$ be such that*

$$\Pr_{(x,y,z) \xleftarrow{\$} X \times Y \times Z} [(x, y, z) \in B] = \varepsilon.$$

For any $\alpha \leq \varepsilon$, define

$$B_\alpha^X = \left\{ (x, y, z) \in X \times Y \times Z : \Pr_{(y',z') \xleftarrow{\$} Y \times Z} \big[(x, y', z') \in B\big] \geq \varepsilon - \alpha \right\}. \tag{4}$$

Then, the following statements hold for any $\alpha \leq \varepsilon$:

(i) $\Pr_{(x,y,z) \xleftarrow{\$} X \times Y \times Z} \big[(x, y, z) \in B_\alpha^X\big] \geq \alpha$

(ii) $\forall (x, y, z) \in B_\alpha^X$: $\Pr_{(y',z') \xleftarrow{\$} Y \times Z} [(x, y', z') \in B] \geq \varepsilon - \alpha$

(iii) $\Pr_{(x,y,z) \xleftarrow{\$} X \times Y \times Z} \big[(x, y, z) \in B_\alpha^X \mid (x, y, z) \in B\big] \geq \alpha/\varepsilon.$

Proof. Follows immediately by setting Y as $Y \times Z$ in the proof of Lemma 3.1. □

In the following, we consider a scenario of splitting at multiple positions. Again, from the algorithmic perspective, we want to quantify the success probability of an algorithm after four consecutive runs. The first run takes random input (x, y, z) and the second run considers the input as (x, y', z'), where $(y', z') \xleftarrow{\$} Y \times Z$ which is basically 1-left splitting. The third and fourth runs, respectively, take random z'' and z''' as inputs with fixed input (x, y). This essentially represents a 2-right splitting. Therefore, we refer to the following lemma as "(1,2)-Multi-Splitting Lemma". Note that this lemma is used only as an intermediary for proving our next result, (2,2)-multi-splitting lemma (Lemma 3.7).

Lemma 3.6 ((1,2)-Multi-Splitting Lemma)**.** *Let X, Y and Z be finite sets. Let $B \subset X \times Y \times Z$ be such that*

$$\Pr_{(x,y,z) \xleftarrow{\$} X \times Y \times Z} [(x, y, z) \in B] = \varepsilon.$$

Then, for any $\alpha \leq \varepsilon$, we have

$$\Pr_{\substack{(x,y,z) \xleftarrow{\$} X \times Y \times Z \\ (y',z') \xleftarrow{\$} Y \times Z \\ z'' \xleftarrow{\$} Z \\ z''' \xleftarrow{\$} Z}} \big[(x, y, z) \in B \wedge (x, y', z') \in B \wedge (x, y, z'') \in B \wedge (x, y, z''') \in B\big] \geq (\varepsilon - \alpha)^3 \cdot \alpha^3/\varepsilon^2.$$

Proof. Let 2-Evt denote the event "$(x,y,z) \in B \wedge (x,y,z'') \in B \wedge (x,y,z''') \in B$". Then, we get,

$$\begin{aligned}
\Delta_1 &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y',z') \in B \mid 2\text{-}\mathsf{Evt}\big]\\
&\geq \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y',z') \in B \wedge (x,y,z) \in B_\alpha^X \mid 2\text{-}\mathsf{Evt}\big]\\
&= \Delta_{11}\cdot\Delta_{12},
\end{aligned}$$

where B_α^X is as defined in Proposition 3.5,

$$\begin{aligned}
\Delta_{11} &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y',z') \in B \mid (x,y,z) \in B_\alpha^X \wedge 2\text{-}\mathsf{Evt}\big]\\
&= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y',z') \in B \mid (x,y,z) \in B_\alpha^X \wedge (x,y,z) \in B \wedge (x,y,z'') \in B \wedge (x,y,z''') \in B\big]\\
&\geq \varepsilon - \alpha \qquad \text{[follows from item (ii) of Proposition 3.5]}
\end{aligned}$$

and

$$\begin{aligned}
\Delta_{12} &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y,z) \in B_\alpha^X \mid 2\text{-}\mathsf{Evt}\big]\\
&= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}} \big[(x,y,z) \in B_\alpha^X \mid (x,y,z) \in B \wedge (x,y,z'') \in B \wedge (x,y,z''') \in B\big] && (5)\\
&\geq \Pr_{(x,y,z)\xleftarrow{\$}X\times Y\times Z} \big[(x,y,z) \in B_\alpha^X \mid (x,y,z) \in B\big] && (6)\\
&\geq \alpha/\varepsilon \qquad \text{[follows from item (iii) of Proposition 3.5].}
\end{aligned}$$

Similarly to Claim 3.2, "$(x,y,z'') \in B \wedge (x,y,z''') \in B$" increases the likelihood of "$(x,y,z) \in B_\alpha^X$". This fact is used in the transition from equation (5) to equation (6).

So, we can write

$$\Delta_1 \geq (\varepsilon - \alpha)\cdot \alpha/\varepsilon. \qquad (7)$$

Finally, we have

$$\begin{aligned}
\Delta &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}}\big[(x,y,z)\in B\wedge(x,y',z')\in B\wedge(x,y,z'')\in B\wedge(x,y,z''')\in B\big]\\
&= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}}\big[(x,y',z')\in B \mid \textsf{2-Evt}\big]\cdot\Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ z''\xleftarrow{\$}Z\\ z'''\xleftarrow{\$}Z}}[\textsf{2-Evt}]\\
&\geq (\varepsilon-\alpha)\cdot(\alpha/\varepsilon)\cdot(\varepsilon-\alpha)^2\cdot\alpha^2/\varepsilon \qquad \text{[using equation (7) and Lemma 3.4]}\\
&= (\varepsilon-\alpha)^3\cdot\alpha^3/\varepsilon^2.
\end{aligned}$$

□

The following lemma is simply an extension of (1,2)-Multi-Splitting Lemma. We, essentially, consider one more 1-left splitting in addition to (1,2)-Multi-Splitting. The lemma is formally stated as follows.

Lemma 3.7 ((2,2)-Multi-Splitting Lemma)**.** *Let X, Y and Z be finite sets. Let $B\subset X\times Y\times Z$ be such that*

$$\Pr_{(x,y,z)\xleftarrow{\$}X\times Y\times Z}[(x,y,z)\in B]=\varepsilon.$$

Then, for any $\alpha\leq\varepsilon$, we have

$$\Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ (y'',z'')\xleftarrow{\$}Y\times Z\\ z'''\xleftarrow{\$}Z\\ z''''\xleftarrow{\$}Z}}\big[(x,y,z)\in B\wedge(x,y',z')\in B\wedge(x,y'',z'')\in B\wedge(x,y,z''')\in B\wedge(x,y,z'''')\in B\big]\geq(\varepsilon-\alpha)^4\cdot\alpha^4/\varepsilon^3.$$

Proof. The proof is given in Section C.2. □

Lemma 3.8 (Sp. (1,2)-Multi-Splitting Lemma)**.** *Let X, Y, Z and U be finite sets. Let $B\subset X\times Y\times Z\times U$ be such that*

$$\Pr_{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U}[(x,y,z,u)\in B]=\varepsilon.$$

Then, for any $\alpha\leq\varepsilon$, we have

$$\Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}}\left[\substack{(x,y,z,u)\in B\wedge(x,y,z',u')\in B\wedge\\ (x,y',z,u'')\in B\wedge(x,y',z',u''')\in B}\right]\geq(\varepsilon-\alpha)^3\cdot\alpha^3/\varepsilon^2.$$

Proof. The proof is given in Section C.1. □

3.3 Single/Multi-Splitting Forking Abstractions

We are now ready to propose three forking abstractions, Lemmas 3.9, 3.11 and 3.13 based on the splitting Lemmas 3.4, 3.7 and 3.8 respectively. As mentioned earlier, an algorithm $\mathcal{B}$ (often referred to as wrapper) which outputs a forking index or forking indices along with some side output (possibly signature(s)) is run several times with related inputs in each of its runs. The output of the wrapper algorithms is, in turn, used in the reduction to solve the underlying hard problem. Here, three types of wrapper algorithm are considered, which output (J, σ), $(J_1, J_2, \sigma_1, \sigma_2)$, and (J_1, J_2, σ), and whose success probabilities depend on the splitting lemmas, Lemmas 3.4, 3.11, and 3.8, respectively. In fact, the forking lemmas basically give the success probability that $\mathcal{B}$ returns the same **fixed** index or indices in all its runs and the $\boldsymbol{h}$-values (involved in these runs) at that index or indices fulfil some restrictions (required by special soundness in Definition 2.13 and 2.14). Finally, Corollaries 3.10, 3.12, and 3.14, respectively, extend the analysis of Lemmas 3.4, 3.11 and 3.8 by considering all indices, instead of fixed index or indices.

Lemma 3.9 (2-Single-Splitting-Forking Lemma)**.** *Fix a positive integer ν and a set R with $|\mathrm{R}| \geq 3$. Also, fix a set Σ of side output, a set $\mathcal{I}$ of instances, and a randomness space Ω. Let $\mathcal{B}$ be an algorithm that, on input $(I, \boldsymbol{h}) \in \mathcal{I} \times \mathrm{R}^\nu$ and randomness $\omega \in \Omega$, outputs a tuple (j, σ), where $j \in [0, \nu]$ and $\sigma \in \Sigma$. For $j \in [\nu]$, define the set $\mathcal{W}_j \subset \mathcal{I} \times \Omega \times \mathrm{R}^\nu$ as*

$$\mathcal{W}_j = \{(I, \omega, \boldsymbol{h}) \in \mathcal{I} \times \Omega \times \mathsf{R}^\nu : \ (j, \sigma) \longleftarrow \mathcal{B}(I, \boldsymbol{h}; \omega)\}.$$

For any $j \in [\nu]$ and $\mathcal{C} \subset \mathcal{W}_j$, define

$$\mathsf{acc}(\mathcal{C}) = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu} [(I, \omega, \boldsymbol{h}) \in \mathcal{C}]$$

and

$$\mathsf{frk}(\mathcal{C}, j) = \Pr_{\substack{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu \\ \boldsymbol{h}' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j-1]} \\ \boldsymbol{h}'' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j-1]}}} \left[{}^{h_j \neq h'_j \neq h''_j \neq h_j \wedge}_{(I,\omega,\boldsymbol{h}) \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}'') \in \mathcal{C}} \right].$$

Then

$$\mathsf{frk}(\mathcal{C}, j) \geq \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^2(\mathcal{C})}{16} - \frac{3}{|\mathrm{R}|} \right).$$

Proof. Let SEvt denote the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}'') \in \mathcal{C}$". Set $X = \mathcal{I} \times \Omega \times \mathrm{R}^{j-1}$ and $Y = \mathrm{R}^{\nu-j+1}$. So, we can write $\mathcal{I} \times \Omega \times \mathsf{R}^\nu = X \times Y$. Applying Lemma 3.4 with $\varepsilon = \mathsf{acc}(\mathcal{C})$ and $\alpha = \varepsilon/2$, we have

$$\Pr_{\substack{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu \\ \boldsymbol{h}' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j-1]} \\ \boldsymbol{h}'' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j-1]}}} [\mathsf{SEvt}] \geq \frac{\mathsf{acc}^3(\mathcal{C})}{16}.$$

Let HEvt denote the event "$h_j \neq h'_j \neq h''_j \neq h_j$". We can then calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk}(\mathcal{C}, j) &= \mathsf{Pr}\left[\mathsf{SEvt} \wedge \mathsf{HEvt}\right] \\
&= \mathsf{Pr}\left[\mathsf{SEvt}\right] - \mathsf{Pr}\left[\mathsf{SEvt} \wedge \overline{\mathsf{HEvt}}\right] \\
&\geq \mathsf{Pr}\left[\mathsf{SEvt}\right] - \mathsf{Pr}\left[(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge \overline{\mathsf{HEvt}}\right] \qquad (8) \\
&\geq \mathsf{acc}^3(\mathcal{C})/16 - \mathsf{Pr}\left[(I, \omega, \boldsymbol{h}) \in \mathcal{C}\right] \cdot \mathsf{Pr}\left[h_j = h'_j \vee h_j = h''_j \vee h'_j = h''_j\right] \qquad (9) \\
&\geq \mathsf{acc}^3(\mathcal{C})/16 - \mathsf{acc}(\mathcal{C}) \cdot 3/|\mathrm{R}| \qquad \text{[using union bound]} \\
&= \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^2(\mathcal{C})}{16} - \frac{3}{|\mathrm{R}|}\right).
\end{aligned}$$

Note that from equation (8) to equation (9), we use the fact that the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C}$" and $\overline{\mathsf{HEvt}}$ are independent as h_j, h'_j and h''_j are chosen independently. This completes the proof. □

The following result is an extension of the original Forking Lemma of [BN06] from one forking to two forkings.

Corollary 3.10. *Define*

$$\mathcal{W} = \bigcup_{j=1}^{\nu} \mathcal{W}_j, \ \mathsf{acc} = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I} \times \Omega \times \mathsf{R}^{\nu}} \left[(I, \omega, \boldsymbol{h}) \in \mathcal{W}\right] \ \textit{and} \ \mathsf{frk} = \sum_{j=1}^{\nu} \mathsf{frk}(\mathcal{W}_j, j).$$

Then, we have

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^2}{16 \cdot \nu^2} - \frac{3}{|\mathrm{R}|}\right).$$

Proof. First, note that $\mathsf{acc} = \sum_{j=1}^{\nu} \mathsf{acc}(\mathcal{W}_j)$. Then, we calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk} &= \sum_{j=1}^{\nu} \mathsf{frk}(\mathcal{W}_j, j) \geq \sum_{j=1}^{\nu} \mathsf{acc}(\mathcal{W}_j) \cdot \left(\frac{\mathsf{acc}^2(\mathcal{W}_j)}{16} - \frac{3}{|\mathrm{R}|}\right) \qquad \text{[using Lemma 3.9]} \\
&= \left(\sum_{j=1}^{\nu} \frac{\mathsf{acc}^3(\mathcal{W}_j)}{16}\right) - \frac{3 \cdot \mathsf{acc}}{|\mathrm{R}|} \geq \frac{1}{16 \cdot \nu^2} \left(\sum_{j=1}^{\nu} \mathsf{acc}(\mathcal{W}_j)\right)^3 - \frac{3 \cdot \mathsf{acc}}{|\mathrm{R}|} \\
&= \frac{\mathsf{acc}^3}{16 \cdot \nu^2} - \frac{3 \cdot \mathsf{acc}}{|\mathrm{R}|} = \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^2}{16 \cdot \nu^2} - \frac{3}{|\mathrm{R}|}\right),
\end{aligned}$$

where we use Proposition A.1 with $a = 3$ in the above intermediate inequality. This completes the proof. □

Lemma 3.11 ((2,2)-Multi-Splitting-Forking Lemma)**.** *Fix a positive integer ν and a set R with $|\mathrm{R}| \geq 6$. Also, fix a set Σ of side output, a set $\mathcal{I}$ of instances, and a randomness space Ω. Let $\mathcal{B}$ be an algorithm that, on input $(I, \boldsymbol{h}) \in \mathcal{I} \times \mathrm{R}^{\nu}$ and randomness $\omega \in \Omega$, outputs a tuple $(j_1, j_2, \sigma_1, \sigma_2)$, where $j_1, j_2 \in [0, \nu]$ and $\sigma_1, \sigma_2 \in \Sigma$. For $j_1, j_2 \in [\nu]$ with $j_1 < j_2$, define the set $\mathcal{W}_{j_1,j_2} \subset \mathcal{I} \times \Omega \times \mathrm{R}^{\nu}$ as*

$$\mathcal{W}_{j_1,j_2} = \{(I, \omega, \boldsymbol{h}) \in \mathcal{I} \times \Omega \times \mathsf{R}^{\nu} : \ (j_1, j_2, \sigma_1, \sigma_2) \longleftarrow \mathcal{B}(I, \boldsymbol{h}; \omega)\}.$$

For any $j_1, j_2 \in [\nu]$ with $j_1 < j_2$ and $\mathcal{C} \subset \mathcal{W}_{j_1,j_2}$, define

$$\mathsf{acc}(\mathcal{C}) = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I} \times \Omega \times \mathsf{R}^{\nu}} \left[(I, \omega, \boldsymbol{h}) \in \mathcal{C}\right]$$

$$\mathsf{frk}(\mathcal{C}, j_1, j_2) = \Pr_{\substack{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu \\ \boldsymbol{h}' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_1-1]} \\ \boldsymbol{h}'' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_1-1]} \\ \boldsymbol{h}''' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_2-1]} \\ \boldsymbol{h}'''' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_2-1]}}} \left[\substack{h_{j_1} \neq h'_{j_1} \neq h''_{j_1} \neq h_{j_1} \wedge h_{j_2} \neq h'''_{j_2} \neq h''''_{j_2} \neq h_{j_2} \wedge \\ (I,\omega,\boldsymbol{h}) \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}'') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}''') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}'''') \in \mathcal{C}} \right].$$

Then

$$\mathsf{frk}(\mathcal{C}, j_1, j_2) \geq \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^4(\mathcal{C})}{256} - \frac{6}{|\mathrm{R}|} \right).$$

Proof. Let SEvt denote the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}'') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}''') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}'''') \in \mathcal{C}$". Set $X = \mathcal{I} \times \Omega \times \mathrm{R}^{j_1-1}$, $Y = \mathrm{R}^{j_2-j_1}$ and $Z = \mathrm{R}^{\nu-j_2+1}$. So, we can write the sample space as $\mathcal{I} \times \Omega \times \mathsf{R}^\nu = X \times Y \times Z$. Applying Lemma 3.7 with $\varepsilon = \mathsf{acc}(\mathcal{C})$ and $\alpha = \varepsilon/2$, we have

$$\Pr_{\substack{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathrm{R}^\nu \\ \boldsymbol{h}' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_1-1]} \\ \boldsymbol{h}'' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_1-1]} \\ \boldsymbol{h}''' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_2-1]} \\ \boldsymbol{h}'''' \xleftarrow{\$} \mathrm{R}^\nu | \boldsymbol{h}_{[j_2-1]}}} [\mathsf{SEvt}] \geq \frac{\mathsf{acc}^5(\mathcal{C})}{256}.$$

Let HEvt denote the event "$h_{j_1} \neq h'_{j_1} \neq h''_{j_1} \neq h_{j_1} \wedge h_{j_2} \neq h'''_{j_2} \neq h''''_{j_2} \neq h_{j_2}$". Then, we can calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk}(\mathcal{C}, j_1, j_2) &= \Pr[\mathsf{SEvt} \wedge \mathsf{HEvt}] \\
&= \Pr[\mathsf{SEvt}] - \Pr\left[\mathsf{SEvt} \wedge \overline{\mathsf{HEvt}}\right] \\
&\geq \Pr[\mathsf{SEvt}] - \Pr\left[(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge \overline{\mathsf{HEvt}}\right] && (10) \\
&\geq \mathsf{acc}^5(\mathcal{C})/256 - \Pr[(I, \omega, \boldsymbol{h}) \in \mathcal{C}] \cdot \Pr\left[\overline{\mathsf{HEvt}}\right] && (11) \\
&\geq \mathsf{acc}^5(\mathcal{C})/256 - \mathsf{acc}(\mathcal{C}) \cdot \Pr\left[\substack{h_{j_1} = h'_{j_1} \vee h_{j_1} = h''_{j_1} \vee h'_{j_1} = h''_{j_1} \vee \\ h_{j_2} = h'''_{j_2} \vee h_{j_2} = h''''_{j_2} \vee h'''_{j_2} = h''''_{j_2}} \right] \\
&\geq \mathsf{acc}^5(\mathcal{C})/256 - \mathsf{acc}(\mathcal{C}) \cdot 6/|\mathrm{R}| \qquad \text{(using union bound)} \\
&= \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^4(\mathcal{C})}{256} - \frac{6}{|\mathrm{R}|} \right).
\end{aligned}$$

Note that from equation (10) to equation (11), we use the fact that the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C}$" and $\overline{\mathsf{HEvt}}$ are independent as $h_{j_1}, h'_{j_1}, h''_{j_1}, h_{j_2}, h'''_{j_2}$ and h''''_{j_2} are chosen independently. This completes the proof. □

Corollary 3.12. *Define*

$$\mathcal{W} = \bigcup_{\substack{j_1, j_2 \in [\nu] \\ j_1 < j_2}} \mathcal{W}_{j_1, j_2}, \ \mathsf{acc} = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathrm{R}^\nu} [(I, \omega, \boldsymbol{h}) \in \mathcal{W}] \ \textit{and} \ \mathsf{frk} = \sum_{\substack{j_1, j_2 \in [\nu] \\ j_1 < j_2}} \mathsf{frk}(\mathcal{W}_{j_1, j_2}, j_1, j_2).$$

Then, we have

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^4}{16 \cdot (\nu^2 - \nu)^4} - \frac{6}{|\mathrm{R}|} \right).$$

Proof. The proof will be found in Section C.3. □

Lemma 3.13 (Special (1,2)-Multi-Splitting-Forking Lemma)**.** *Fix a positive integer ν and a set* R *with* $|\mathrm{R}| \geq 2$. *Also, fix a set Σ of side outputs, a set $\mathcal{I}$ of instances, and a randomness space Ω. Let $\mathcal{B}$ be an algorithm that, on input $(I, \boldsymbol{h}) \in \mathcal{I} \times \mathrm{R}^\nu$ and randomness $\omega \in \Omega$, outputs a tuple (j_1, j_2, σ), where $j_1, j_2 \in [0, \nu]$ and $\sigma \in \Sigma$. For $j_1, j_2 \in [\nu]$ with $j_1 < j_2$, define the set $\mathcal{W}_{j_1,j_2} \subset \mathcal{I} \times \Omega \times \mathrm{R}^\nu$ as*

$$\mathcal{W}_{j_1,j_2} = \{(I, \omega, \boldsymbol{h}) \in \mathcal{I} \times \Omega \times \mathsf{R}^\nu : \ (j_1, j_2, \sigma) \longleftarrow \mathcal{B}(I, \boldsymbol{h}; \omega)\}.$$

For any $j_1, j_2 \in [\nu]$ with $j_1 < j_2$ and $\mathcal{C} \subset \mathcal{W}_{j_1,j_2}$, define

$$\mathsf{acc}(\mathcal{C}) = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu} [(I, \omega, \boldsymbol{h}) \in \mathcal{C}]$$

$$\mathsf{frk}(\mathcal{C}, j_1, j_2) = \Pr_{\substack{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu \\ \boldsymbol{h}' \longleftarrow \mathrm{R}^\nu | \boldsymbol{h}_{[j_2-1]} \\ \boldsymbol{h}'' \longleftarrow \mathrm{R}^\nu | \boldsymbol{h}_{[j_1-1]} \ s.t\ h''_{j_2} = h_{j_2} \\ \boldsymbol{h}''' \longleftarrow \mathrm{R}^\nu | \boldsymbol{h}''_{[j_2-1]} \ s.t\ h'''_{j_2} = h'_{j_2}}} \left[\substack{h'_{j_2} \neq h_{j_2} \wedge h''_{j_1} \neq h_{j_1} \wedge \\ (I,\omega,\boldsymbol{h}) \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}'') \in \mathcal{C} \wedge (I,\omega,\boldsymbol{h}''') \in \mathcal{C}} \right]. \tag{12}$$

Then

$$\mathsf{frk}(\mathcal{C}, j_1, j_2) \geq \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^3(\mathcal{C})}{64} - \frac{2}{|\mathrm{R}|} \right).$$

Proof. The proof is given in Section C.4. □

Remark 3.1. Note that while sampling $\boldsymbol{h}''$ and $\boldsymbol{h}'''$ in equation (12), we keep your choices $h''_{j_2} = h_{j_2}$ and $h'''_{j_2} = h'_{j_2}$. This will be used in the proof of Theorem 4.2 to ensure the requirement on the second challenge of 5-special soundness (Definition 2.14) of the underlying 5-pass IDS.

Corollary 3.14. *Define*

$$\mathcal{W} = \bigcup_{\substack{j_1,j_2 \in [\nu] \\ j_1 < j_2}} \mathcal{W}_{j_1,j_2}, \ \mathsf{acc} = \Pr_{(I,\omega,\boldsymbol{h}) \xleftarrow{\$} \mathcal{I}\times\Omega\times\mathsf{R}^\nu} [(I, \omega, \boldsymbol{h}) \in \mathcal{W}] \ \textit{and}\ \mathsf{frk} = \sum_{\substack{j_1,j_2 \in [\nu] \\ j_1 < j_2}} \mathsf{frk}(\mathcal{W}_{j_1,j_2}, j_1, j_2).$$

Then, we have

$$\mathsf{frk} \geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^3}{8 \cdot (\nu^2 - \nu)^3} - \frac{2}{|\mathrm{R}|} \right).$$

Proof. The proof is given in Section C.5 □

4 A Modular Security Treatment of MQDSS

In this section, we provide concrete security reduction of 3/5-pass MQDSS in the random oracle model. The 5-pass MQDSS [CHR$^+$16] is obtained by applying the Fiat-Shamir transform [FS86] to r parallel instances of the 5-pass IDS [SSH11]. The authors also presented a security reduction for 5-pass MQDSS, which is qualitative and thus provides only asymptotic assurance. In fact, the paper [CHR$^+$16] itself explicitly states that the derivation of an exact probability bound leads only to a complicated statement.

According to [CHR$^+$16], the reason for considering 5-pass, instead of 3-pass IDS in MQDSS is to reduce the number of rounds, thus gaining efficiency. However, the use of 5-pass IDS leads to an interesting and effective attack [KZ20] on the concrete parameter choices of 5-pass MQDSS that was submitted for NIST PQC standardisation [NIS19]. To prevent this attack, the number of rounds in the underlying IDS should be increased. As observed in [KZ20] due to their attack, 5-pass MQDSS loses its efficiency advantage over the 3-pass variant. Note that the attack on 5-pass MQDSS works due to the lack of tightness in the security reduction. Therefore, for a fair comparison between these two versions of MQDSS, one must take into account the tightness of the reductions. This motivates us to investigate concrete security reductions for 3/5-pass MQDSS. In Section 4.1, we provide an outline of 3-pass MQDSS followed by its concrete security reduction. We then give an outline of 5-pass MQDSS and its concrete security reduction in Section 4.2 followed by a comparative analysis of the relative security of MQDSS variants in Section 4.3.

4.1 3-Pass MQDSS

In this section, we first describe the MQDSS signature based on 3-pass IDS. The description is similar to the 5-pass MQDSS [CHR+16], except that the number of passes is reduced. Let $\mathsf{IDS} = (\mathsf{IDS.KeyGen}, \mathsf{P}, \mathsf{V})$ be the 3-pass IDS of Sakumoto et al. [SSH11] (see Figure 4), where the underlying challenge space is $\mathsf{ChS} = \{0, 1, 2\}$. The three subroutines of the 3-pass MQDSS are given as follows. We use the notation IDS^r for r-parallel runs of IDS (see Remark 2.6).

1. $\mathsf{KeyGen}(\kappa)$. Run $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{IDS.KeyGen}(\kappa)$. Choose a number $r \in \mathbb{N}$ such that $(2/3)^r = \mathsf{negl}(\kappa)$. This choice essentially makes the knowledge error of IDS^r negligible thanks to Lemma B.3. Let $\mathcal{H} : \{0,1\}^* \to \{0,1,2\}^r$ be a collision resistant hash function. Similarly to pk, $\mathcal{H}$ and r are also publicly known. Note that pk implicitly contains the public key ck of the underlying commitment scheme.

2. $\mathsf{Sign}(\mathsf{sk}, \mathrm{M})$. The steps of signature generation are given below.

   (a) run $\mathbf{ct} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_r) \longleftarrow \mathsf{P}_0^r(\mathsf{sk})$

   (b) compute $\mathbf{ch} := \mathcal{H}(\mathrm{M}, \mathbf{ct})$

   (c) run $\mathbf{rs} := (\mathsf{rs}_1, \ldots, \mathsf{rs}_r) \longleftarrow \mathsf{P}_1^r(\mathsf{sk}, \mathbf{ct}, \mathbf{ch})$

   (d) return $\sigma := (\mathbf{ct}, \mathbf{rs})$

3. $\mathsf{Ver}(\mathsf{pk}, \mathrm{M}, \sigma)$. Verification has the following steps.

   (a) compute $\mathbf{ch} := \mathcal{H}(\mathrm{M}, \mathbf{ct})$

   (b) return $\mathsf{V}^r(\mathsf{pk}, \mathbf{ct}, \mathbf{ch}, \mathbf{rs})$

Next, we provide a concrete security reduction for the above construction. Note that by utilising HVZK of the underlying 3-pass IDS (see Lemma B.1), an EUF-CMA forger can be reduced to an EUF-NMA forger. This is a common technique widely used in the security reduction of signature scheme derived from an identification scheme (for example, see [CHR+16, Lemma 4.12] in the context of 5-pass MQDSS). Basically, one has to ensure that an EUF-NMA forger is capable of answering signature queries. Informally, the EUF-NMA forger synthesises a signature by generating r simulated proofs using HVZK of the underlying IDS followed by programming the random oracle for $\mathcal{H}$ at some arguments using r challenges involved in the simulated proofs. If the oracle values of the arguments are already defined by its challenger, then EUF-NMA forger aborts. It can be shown that the probability of not aborting is overwhelming due to the choice of r. Since, for this conversion, the advantage gets reduced by at most a negligible amount, we only focus on the EUF-NMA security of 3/5-pass MQDSS. We point out that due to the application of our modular forking analysis (see Corollary 3.10), the reduction appears significantly simpler and is reminiscent of the original Bellare-Neven forking [BN06].

Theorem 4.1. *Suppose there exists an adversary $\mathcal{A}$ who can break the EUF-NMA security of the 3-pass MQDSS in the random oracle model. Then, using $\mathcal{A}$ as a subroutine, we can create PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for breaking the MQ problem (c.f., Definition 2.15) and the computational binding property of the underlying commitment scheme respectively with the following relation of the respective advantages:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-NMA}}(\kappa) \quad \leq \quad \left(16 \cdot \nu^2 \cdot \left(\mathsf{Adv}_{\mathcal{B}_1}^{\text{MQ}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_2}^{\text{Binding}}(\kappa)\right) / f\right)^{1/3}$$

where ν is the number of hash queries and $f = 1 - (7/9)^r$.

Proof. Let $\mathcal{A}$ be an NMA-attacker that produces a forgery after making at most ν many random oracle queries. Then, we break either the MQ problem or the computational binding property of the underlying commitment scheme using $\mathcal{A}$ as a subroutine. Next, we define our wrapper algorithm.

Wrapper Algorithm. Let Mq denote the set of all MQ problem instances. The wrapper algorithm $\mathcal{B}$ (defined as Algorithm 1) takes $(\mathsf{inst}, \boldsymbol{h}) \in \mathsf{Mq} \times \mathsf{R}^\nu$ as input and creates an environment for $\mathcal{A}$, where $\mathsf{R} = \{0, 1, 2\}^r$. Throughout our analysis, $1/|\mathsf{R}| = \mathsf{negl}(\kappa)$ due to the choice of r. When $\mathcal{A}$ returns a message-signature pair as a forgery, $\mathcal{B}$ returns a pair (J, σ), where $J \in [0, \nu]$ is called the forking index and σ is called side output. The random oracle is implemented using the random vector $\boldsymbol{h}$ as follows. For a query argument arg, return $\mathcal{H}(\mathsf{arg})$, if it is defined, i.e., arg was queried earlier. Otherwise, $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$, set $\mathcal{H}(\mathsf{arg}) = h_{\mathsf{ctr}}$, update $\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}(\mathsf{arg}))\}$ and return $\mathcal{H}(\mathsf{arg})$. We say that the wrapper $\mathcal{B}$ is successful if the output forking index J is not equal to 0. Let acc denote the success probability of $\mathcal{B}$. Then, we can write,

$$\begin{aligned} \mathsf{acc} \quad &\geq \quad (1 - \frac{1}{|\mathsf{R}|}) \cdot \varepsilon \\ &\approx \quad \varepsilon \end{aligned} \tag{13}$$

where $\varepsilon = \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-NMA}}(\kappa)$. Note that the term $(1 - 1/|\mathsf{R}|)$ in the above expression arises from steps 6 to 8 of Algorithm 1, as $\mathcal{A}$ may produce a valid forgery by guessing the hash value $\mathcal{H}(\mathrm{M}, \mathbf{ct})$ with at most a uniform probability $1/|\mathsf{R}|$. Let Ω be the domain from which $\mathcal{A}$ samples its randomness. Note that following the notation $\mathcal{W}$ and $\mathcal{W}_j$ developed in Lemma 3.9 and Corollary 3.10, we can also express acc as follows.

$$\mathsf{acc} = \Pr_{(\mathsf{inst}, \omega, \boldsymbol{h}) \xleftarrow{\$} \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu} [(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathcal{W}],$$

where $\mathcal{W} = \{(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu : J \geq 1;\ (J, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)\}$. It is worth mentioning that the wrapper algorithm $\mathcal{B}$ does not sample any randomness. Moreover, the supplied random coin ω to $\mathcal{B}$ actually serves as the internal randomness for $\mathcal{A}$. We can write $\mathcal{W} = \cup_{j=1}^{\nu} \mathcal{W}_j$, where $\mathcal{W}_j = \{(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu :\ (j, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)\}$ for $j \in [\nu]$.

Algorithm 1 Wrapper for NMA

```
 1: procedure $\mathcal{B}(\mathsf{inst}, \boldsymbol{h})$
 2:     run $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(\kappa)$
 3:     pick $r \in \mathbb{N}$ such that $(2/3)^r = \mathsf{negl}(\kappa)$
 4:     $\mathsf{List} \leftarrow \emptyset$ and $\mathsf{ctr} \leftarrow 0$        ▷ List stores tuples of the form $(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}(\mathsf{arg}))$
 5:     $(\mathrm{M}, (\mathbf{ct}, \mathbf{rs})) \longleftarrow \mathcal{A}^{\mathcal{O}_\mathcal{H}}(\mathsf{inst}, \mathsf{ck}, r)$
 6:     if $(*, (\mathrm{M}, \mathbf{ct}), *) \notin \mathsf{List}$ then
 7:         return $(0, \epsilon)$
 8:     end if
 9:     let $J \in [\nu]$ such that $(J, (\mathrm{M}, \mathbf{ct}), \mathbf{ch}) \in \mathsf{List}$
10:     set $\sigma = (\mathbf{ct}, \mathbf{ch}, \mathbf{rs})$
11:     if $\mathsf{Ver}(\mathrm{M}, \sigma, \mathsf{pk}) = 0$ then        ▷ where $\mathsf{pk} = (\mathsf{inst}, \mathsf{ck})$
12:         return $(0, \epsilon)$
13:     end if
14:     return $(J, \sigma)$
15: end procedure
```

Forking Algorithm. The forking steps are described in Algorithm 2. Note that it involves one additional forking compared to the general forking algorithm of Bellare-Neven [BN06]. The success probability of $\mathsf{F}_{\mathcal{B}}$ can be expressed in terms of the forking probability frk as defined in Corollary 3.10. In fact,

$$\begin{aligned}
\mathsf{Succ}(\mathsf{F}_{\mathcal{B}}) &= \Pr\left[b = 1 : \ \mathsf{inst} \xleftarrow{\$} \mathsf{Mq}; \ (b, \sigma, \sigma', \sigma'') \longleftarrow \mathsf{F}_{\mathcal{B}}(\mathsf{inst})\right] \\
&= \sum_{j=1}^{\nu} \mathsf{frk}(\mathcal{W}_j, j) \\
&= \mathsf{frk} \\
&\geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^2}{16 \cdot \nu^2} - \frac{3}{|\mathsf{R}|}\right) \qquad \text{[by Corollary 3.10]} \\
&\geq \varepsilon \cdot \left(\frac{\varepsilon^2}{16 \cdot \nu^2} - \frac{3}{|\mathsf{R}|}\right) \qquad \text{[using equation (13)]} \\
&\approx \frac{\varepsilon^3}{16 \cdot \nu^2} \qquad (14)
\end{aligned}$$

where we refer to Lemma 3.9 for the definition of $\mathsf{frk}(\mathcal{W}_j, j)$.

Algorithm 2 Forking for NMA

```
1: procedure $\mathsf{F}_{\mathcal{B}}(\mathsf{inst})$
2:     choose $\boldsymbol{h} \xleftarrow{\$} (\{0,1,2\}^r)^\nu$
3:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}$
4:     $(J, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)$        ▷ 1st run
5:     if $J = 0$ then
6:         return $(0, \epsilon, \epsilon, \epsilon)$
7:     end if
8:     pick $h'_J, \ldots, h'_\nu \xleftarrow{\$} \{0,1,2\}^r$
9:     set $\boldsymbol{h}' = (h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_\nu)$
10:    $(J', \sigma') \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}'; \omega)$        ▷ 2nd run
11:    if $J \neq J'$ or $h_J = h'_J$ then
12:        return $(0, \epsilon, \epsilon, \epsilon)$
13:    end if
14:    pick $h''_J, \ldots, h''_\nu \xleftarrow{\$} \{0,1,2\}^r$
15:    set $\boldsymbol{h}'' = (h_1, \ldots, h_{J-1}, h''_J, \ldots, h''_\nu)$
16:    $(J'', \sigma'') \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}''; \omega)$        ▷ 3rd run
17:    if $J' \neq J''$ or $h_J = h''_J$ or $h'_J = h''_J$ then
18:        return $(0, \epsilon, \epsilon, \epsilon)$
19:    end if
20:    return $(1, \sigma, \sigma', \sigma'')$
21: end procedure
```

Algorithm 3 MQ Solver

```
1: procedure $\mathsf{Simu}(\mathcal{P}, \boldsymbol{y}^*)$        ▷ output $\boldsymbol{x}^* \in \mathbb{F}^n$ such that $\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^*$
2:     set $\mathsf{inst} = (\mathcal{P}, \boldsymbol{y}^*)$
3:     $(b, \sigma, \sigma', \sigma'') \longleftarrow \mathsf{F}_{\mathcal{B}}(\mathsf{inst})$
4:     if $b = 0$ then
5:         abort
6:     end if
7:     parse $\sigma$ as $(\mathbf{ct}, \mathbf{ch}, \mathbf{rs})$
8:     parse $\sigma'$ as $(\mathbf{ct}, \mathbf{ch}', \mathbf{rs}')$
9:     parse $\sigma''$ as $(\mathbf{ct}, \mathbf{ch}'', \mathbf{rs}'')$
10:    find an $i \in [r]$ such that $\mathsf{ch}_i \neq \mathsf{ch}'_i \neq \mathsf{ch}''_i \neq \mathsf{ch}_i$.
11:    if such $i$ is not found then
12:        abort
13:    end if
14:    set $\pi = (\mathsf{ct}_i, \mathsf{ch}_i, \mathsf{rs}_i)$
15:    set $\pi' = (\mathsf{ct}_i, \mathsf{ch}'_i, \mathsf{rs}'_i)$
16:    set $\pi'' = (\mathsf{ct}_i, \mathsf{ch}''_i, \mathsf{rs}''_i)$
17:    $\boldsymbol{x}^* \longleftarrow \mathsf{Extr}(\pi, \pi', \pi'')$        ▷ Such an extractor exists due to Lemma B.2
18:    return $\boldsymbol{x}^*$
19: end procedure
```

MQ Solver. The MQ problem solver Simu is described in Algorithm 3. It takes an MQ problem instance $(\mathcal{P}, \boldsymbol{y}^*)$ as input and runs $\mathsf{F}_{\mathcal{B}}$. Recall that with probability $\mathsf{Succ}(\mathsf{F}_{\mathcal{B}})$, the forking algorithm returns three signatures $\sigma, \sigma', \sigma''$ for the same message. Using Proposition A.2, we get the probability of reaching step 17 of Algorithm 3 without abort to be $\mathsf{Succ}(\mathsf{F}_{\mathcal{B}}) \cdot (1 - (7/9)^r) = \mathsf{Succ}(\mathsf{F}_{\mathcal{B}}) \cdot \mathsf{f}$, where $\mathsf{f} = 1 - (7/9)^r$. Note that one can make f negligibly close to 1 by suitably choosing the value of r. If the underlying commitment scheme is computationally binding, then $\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^*$, where $\boldsymbol{x}^*$ is the output of the extractor in step 17. Therefore, the success probability of finding a valid solution is

$$\begin{aligned} \mathsf{Adv}^{\text{MQ}}_{\mathsf{Simu}}(\kappa) &\geq \mathsf{Succ}(\mathsf{F}_{\mathcal{B}}) \cdot \mathsf{f} - \mathsf{Adv}^{\text{Binding}}_{\mathsf{Simu}}(\kappa) \\ &\geq \frac{\varepsilon^3}{16 \cdot \nu^2} \cdot \mathsf{f} - \mathsf{Adv}^{\text{Binding}}_{\mathsf{Simu}}(\kappa), \end{aligned} \tag{15}$$

where in the last step, we use equation (14). This completes the proof. □

Since the underlying commitment scheme is computational binding, one can approximate equation (15) as

$$\varepsilon \leq \nu^{2/3} \cdot \left(\mathsf{Adv}^{\text{MQ}}_{\mathsf{Simu}}(\kappa)\right)^{1/3} \tag{16}$$

where we assume that f is negligibly close to 1.

4.2 5-Pass MQDSS

We start with an outline of 5-pass MQDSS [CHR$^+$16] based on the Fiat-Shamir transform [FS86] on 5-pass IDS [SSH11] as appeared in [CHR$^+$16, Construction 4.7]. We refer to [CHR$^+$16] for further details on MQDSS. Let $\mathsf{IDS} = (\mathsf{IDS.KeyGen}, \mathsf{P}, \mathsf{V})$ be the 5-pass IDS of Sakumoto et al. [SSH11] (see Figure 5), where the first and second challenge spaces are $\mathsf{ChS}_1 = \mathbb{F}$ and $\mathsf{ChS}_2 = \{0, 1\}$ respectively. The three subroutines of the MQDSS are as follows.

1. $\mathsf{KeyGen}(\kappa)$. Run $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{IDS.KeyGen}(\kappa)$. Choose a number $r \in \mathbb{N}$ such that $(\frac{1}{2} + \frac{1}{2q})^r = \mathsf{negl}(\kappa)$. This choice essentially makes the knowledge error of IDS^r negligible thanks to Lemma B.6. Let $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{F}^r$ and $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^r$ be collision-resistant hash functions. Similarly to pk, $\mathcal{H}_1, \mathcal{H}_2$ and r are also publicly known. Note that pk implicitly contains the public key ck of the underlying commitment scheme.
2. $\mathsf{Sign}(\mathsf{sk}, \mathrm{M})$. The steps of signature generation are given below.
   (a) run $\mathbf{ct} := (\mathsf{ct}_1, \ldots, \mathsf{ct}_r) \longleftarrow \mathsf{P}^r_0(\mathsf{sk})$
   (b) compute $\mathbf{ch}_1 := \mathcal{H}_1(\mathrm{M}, \mathbf{ct})$
   (c) run $\mathbf{rs}_1 := (\mathsf{rs}_{1,1}, \ldots, \mathsf{rs}_{1,r}) \longleftarrow \mathsf{P}^r_1(\mathsf{sk}, \mathbf{ct}, \mathbf{ch}_1)$
   (d) compute $\mathbf{ch}_2 := \mathcal{H}_2(\mathrm{M}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1)$
   (e) run $\mathbf{rs}_2 := (\mathsf{rs}_{2,1}, \ldots, \mathsf{rs}_{2,r}) \longleftarrow \mathsf{P}^r_2(\mathsf{sk}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \mathbf{ch}_2)$
   (f) return $\sigma = (\mathbf{ct}, \mathbf{rs}_1, \mathbf{rs}_2)$
3. $\mathsf{Ver}(\mathsf{pk}, \mathrm{M}, \sigma)$. The verification steps are given below.
   (a) compute $\mathbf{ch}_1 := \mathcal{H}_1(\mathrm{M}, \mathbf{ct})$ and $\mathbf{ch}_2 := \mathcal{H}_2(\mathrm{M}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1)$
   (b) return $\mathsf{V}^r(\mathsf{pk}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \mathbf{ch}_2, \mathbf{rs}_2)$

Next, we show a concrete security reduction of the 5-pass MQDSS. Similarly to the 3-pass MQDSS, we only address EUF-NMA security. Unlike the reduction of the 3-pass MQDSS, here we consider some encoding functions to answer the random oracle queries corresponding to two different hash functions using a single random vector.

Theorem 4.2. *Suppose there exists an adversary $\mathcal{A}$ who can break the EUF-NMA security of the 5-pass MQDSS in the random oracle model. Then, using $\mathcal{A}$ as a subroutine, we can create PPT algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for breaking the MQ problem (c.f., Definition 2.15) and the computational binding property of the underlying commitment scheme respectively with the following relation of the respective advantages:*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{EUF-NMA}}(\kappa) \quad \leq \quad \left(8 \cdot (\nu^2 - \nu)^4 \cdot \left(\mathsf{Adv}_{\mathcal{B}_1}^{\text{MQ}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_2}^{\text{Binding}}(\kappa)\right)\right)^{1/4}$$

where ν is the number of hash queries.

Proof. Recall that there are two hash functions $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{F}^r$ and $\mathcal{H}_2 : \{0,1\}^* \to \{0,1\}^r$ which are treated as random oracles. Since the range $\mathbb{F}^r$ (resp. $\{0,1\}^r$) is a finite set, any element of it can be uniquely expressed by the index of a suitably chosen enumeration or index set. In fact, let $\mathsf{R}_1 = \{1, \ldots, q^r\}$ (resp. $\mathsf{R}_2 = \{1, \ldots, 2^r\}$) be the corresponding enumeration. If $\mathsf{Enu}_1 : \mathsf{R}_1 \to \mathbb{F}^r$ (resp. $\mathsf{Enu}_2 : \mathsf{R}_2 \to \{0,1\}^r$) is a one-to-one function, then the i -th element in $\mathbb{F}^r$ (resp. $\{0,1\}^r$) is represented by $\mathsf{Enu}_1(i)$ (resp. $\mathsf{Enu}_2(i)$). W.l.o.g, we assume that such functions exist and are efficiently computable. Consider another set of indexes $\mathsf{R} = \{1, 2, \ldots, \varphi\}$, where $\varphi = q^r \cdot 2^r$. Next, define two encoding functions $\mathsf{Enc}_1 : \mathsf{R} \to \mathsf{R}_1$ and $\mathsf{Enc}_2 : \mathsf{R} \to \mathsf{R}_2$ as follows.

$$\mathsf{Enc}_1(x) = \left\lceil \frac{x}{2^r} \right\rceil \text{ and } \mathsf{Enc}_2(x) = \left\lceil \frac{x}{q^r} \right\rceil \text{ for } x \in \mathsf{R}.$$

These encoding functions help to simulate two different random oracles using only a single random vector. We now list some properties of the encoding functions defined above.

1. $\mathsf{Enc}_1 : \mathsf{R} \to \mathsf{R}_1$ and $\mathsf{Enc}_2 : \mathsf{R} \to \mathsf{R}_2$ are regular.[5]
2. $\mathsf{Enu}_1 \circ \mathsf{Enc}_1 : \mathsf{R} \to \mathbb{F}^r$ and $\mathsf{Enu}_2 \circ \mathsf{Enc}_2 : \mathsf{R} \to \{0,1\}^r$ are also regular.
3. All these functions are efficiently computable.
4. When $x \xleftarrow{\$} \mathsf{R}$, then $(\mathsf{Enu}_1 \circ \mathsf{Enc}_1)(x)$ and $(\mathsf{Enu}_2 \circ \mathsf{Enc}_2)(x)$ are uniformly distributed over $\mathbb{F}^r$ and $\{0,1\}^r$, respectively, due to property (2).

Let $\mathcal{A}$ be an NMA attacker that produces a forgery after making at most ν many random oracle queries (to $\mathcal{H}_1$ and $\mathcal{H}_2$). Then, we break either the MQ problem or the computational binding of the underlying commitment using $\mathcal{A}$ as a subroutine. Following the style of our earlier reduction, we define a wrapper algorithm as follows.

Wrapper Algorithm. Let Mq denote the set of all MQ problem instances. The wrapper algorithm $\mathcal{B}$ (defined as Algorithm 4) takes $(\mathsf{inst}, \boldsymbol{h}) \in \mathsf{Mq} \times \mathsf{R}^\nu$ as input and creates an environment for $\mathcal{A}$. Throughout our analysis, $1/|\mathsf{R}| = \mathsf{negl}(\kappa)$ due to the choice of r. When $\mathcal{A}$ returns a message-signature pair as forgery, $\mathcal{B}$ returns a tuple (J_1, J_2, σ), where $J_1, J_2 \in [0, \nu]$ are called forking indices. The wrapper $\mathcal{B}$ handles the random oracle queries as given below.

```
$\mathcal{O}_{\mathcal{H}_i}(\mathsf{arg})$: // for $i = 1, 2$
1: if $\mathcal{H}_i(\mathsf{arg})$ is not defined then
2:     $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
3:     $\mathcal{H}_i(\mathsf{arg}) \leftarrow (\mathsf{Enu}_i \circ \mathsf{Enc}_i)\,(h_{\mathsf{ctr}})$
4:     $\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))\}$
5: end if
6: return $\mathcal{H}_i(\mathsf{arg})$
```

[5]A function is called regular if all the elements in the range have exactly the same number of preimages. For example, any bijective function is regular.

Algorithm 4 Wrapper for NMA

```
1: procedure $\mathcal{B}(\mathsf{inst}, \boldsymbol{h})$
2:     run $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(\kappa)$
3:     pick $r \in \mathbb{N}$ such that $(\frac{1}{2} + \frac{1}{2q})^r = \mathsf{negl}(\kappa)$
4:     $\mathsf{List} \leftarrow \emptyset$ and $\mathsf{ctr} \leftarrow 0$      ▷ List stores tuples of the form $(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))$ for $i \in [2]$
5:     $(\mathrm{M}, (\mathbf{ct}, \mathbf{rs}_1, \mathbf{rs}_2)) \longleftarrow \mathcal{A}^{\mathcal{O}_{\mathcal{H}_1}, \mathcal{O}_{\mathcal{H}_2}}(\mathsf{inst}, \mathsf{ck}, r)$
6:     if $(*, (\mathrm{M}, \mathbf{ct}), *) \notin \mathsf{List}$ then
7:         return $(0, 0, \epsilon)$
8:     end if
9:     let $J_1 \in [\nu]$ such that $(J_1, (\mathrm{M}, \mathbf{ct}), \mathbf{ch}_1) \in \mathsf{List}$
10:    if $(*, (\mathrm{M}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1), *) \notin \mathsf{List}$ then
11:        return $(0, 0, \epsilon)$
12:    end if
13:    let $J_2 \in [\nu]$ such that $(J_2, (\mathrm{M}, \mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1), \mathbf{ch}_2) \in \mathsf{List}$
14:    set $\sigma = (\mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \mathbf{ch}_2, \mathbf{rs}_2)$
15:    if $\mathsf{Ver}(\mathrm{M}, \sigma, \mathsf{pk}) = 0$ then      ▷ where $\mathsf{pk} = (\mathsf{inst}, \mathsf{ck})$
16:        return $(0, 0, \epsilon)$
17:    end if
18:    return $(J_1, J_2, \sigma)$
19: end procedure
```

By the definition of $\mathcal{B}$, the indices J_1 and J_2 involved in the output can be both zero or non-zero. Note that when they are non-zero, then $J_1 < J_2$ as the input of $\mathcal{H}_2$ includes the output of $\mathcal{H}_1$. We say that the wrapper $\mathcal{B}$ is successful if both forking indices in the output are non-zero.

Let Ω be the domain from which $\mathcal{A}$ samples its randomness. Let $\mathcal{W} = \{(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu : J_1 < J_2;\ (J_1, J_2, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)\}$ and $\mathcal{W}_{j_1,j_2} = \{(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu :\ (j_1, j_2, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)\}$ for $j_1, j_2 \in [\nu]$. Then $\mathcal{W} = \bigcup_{\substack{j_1,j_2 \in [\nu] \\ j_1 < j_2}} \mathcal{W}_{j_1,j_2}$. If acc denotes the success probability of $\mathcal{B}$, then we can write

$$\begin{aligned} \mathsf{acc} &= \Pr_{(\mathsf{inst},\omega,\boldsymbol{h}) \xleftarrow{\$} \mathsf{Mq} \times \Omega \times \mathsf{R}^\nu} [(\mathsf{inst}, \omega, \boldsymbol{h}) \in \mathcal{W}] \\ &\geq (1 - \frac{2}{|\mathsf{R}|}) \cdot \varepsilon \\ &\approx \varepsilon \end{aligned} \tag{17}$$

where $\varepsilon = \mathsf{Adv}_{\mathcal{A}}^{\text{EUF-NMA}}(\kappa)$ and the term $(1 - 2/|\mathsf{R}|)$ involved in above expression is due to steps 6 to 12 of Algorithm 4, which basically corresponds to guessing the output of the random oracles for $\mathcal{H}_1$ and $\mathcal{H}_2$.

Extended Forking Algorithm. The extended[6] forking algorithm is given as Algorithm 5. It takes $\mathsf{inst} \in \mathsf{Mq}$ as input and runs $\mathcal{B}$ four times on related inputs to obtain four signatures $(\sigma, \sigma', \sigma'', \sigma''')$. In the first run, on input $(\mathsf{inst}, \boldsymbol{h}, \omega)$, $\mathcal{B}$ outputs (J_1, J_2, σ). In the second run, $\mathcal{B}$ is rewound at J_2 to get (J_1', J_2', σ'). The second argument for $\mathcal{B}$ in the second run is $\boldsymbol{h}'$. Next, $\mathcal{B}$ is rewound at J_1 (i.e., third run) such that $h''_{J_2} = h_{J_2}$ to get (J_1'', J_2'', σ''), where $\boldsymbol{h}''$ is the second argument of $\mathcal{B}$. Finally, $\mathcal{B}$ is rewound at J_2 (i.e., fourth run) such that $h'''_{J_2} = h'_{J_2}$ to get $(J_1''', J_2''', \sigma''')$, where $\boldsymbol{h}'''$ is the 2nd argument of $\mathcal{B}$. Note that the values h''_{J_2} and h'''_{J_2} are chosen in the above form mainly to fulfil some requirements on the second challenge involved in the definition of $q2$-extractor (c.f., Definition 2.14). If all the requirements are fulfilled by the outputs, then $(\sigma, \sigma', \sigma'', \sigma''')$ are four valid signatures for the same message. We emphasise that here we rewind $\mathcal{B}$ at two indices that correspond to two different hash functions. Let us parse the four signatures as $\sigma = (\mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \mathbf{ch}_2, \mathbf{rs}_2)$, $\sigma' = (\mathbf{ct}', \mathbf{ch}_1', \mathbf{rs}_1', \mathbf{ch}_2', \mathbf{rs}_2')$, $\sigma'' = (\mathbf{ct}'', \mathbf{ch}_1'', \mathbf{rs}_1'', \mathbf{ch}_2'', \mathbf{rs}_2'')$ and $\sigma''' = (\mathbf{ct}''', \mathbf{ch}_1''', \mathbf{rs}_1''', \mathbf{ch}_2''', \mathbf{rs}_2''')$. Note that one can easily verify that

$$\mathbf{ct} = \mathbf{ct}' = \mathbf{ct}'' = \mathbf{ct}'''. \tag{18}$$

Further, the values $h'_{J_1} = h_{J_1}$, $h''_{J_2} = h_{J_2}$, and $h'''_{J_1} = h''_{J_1}$ and $h'''_{J_2} = h'_{J_2}$ set in steps 9, 15 and 21 of Algorithm 5 respectively, and the conditions on h'_{J_2} and h''_{J_1} involved in steps 11, 17 imply that

$$\mathbf{ch}_1 = \mathbf{ch}_1' \quad \neq \quad \mathbf{ch}_1'' = \mathbf{ch}_1''' \tag{19}$$

$$\mathbf{ch}_2 = \mathbf{ch}_2'' \quad \neq \quad \mathbf{ch}_2' = \mathbf{ch}_2'''. \tag{20}$$

[6]We use the term "extended" because here the forking algorithm uses some encoding functions which is different from the usual forking algorithm, e.g., [BN06].

We remark that the conditions involved in the boxes in steps 11 and 17 of Algorithm 5 are redundant. In fact, $J_2 = J_2'$ (resp. $J_2'' = J_2'''$) implies that $J_1 = J_1'$ (resp. $J_1'' = J_1'''$) due to the event "the input and the output of $\mathcal{H}_1$ appear as the input of $\mathcal{H}_2$ for a valid signature". The reason for keeping them in place is simply to follow Corollary 3.14.

Algorithm 5 Extended Forking for NMA

```
1: procedure $\mathsf{ExtF}_{\mathcal{B}}(\mathsf{inst})$
2:     choose $\boldsymbol{h} \xleftarrow{\$} \mathsf{R}^{\nu}$
3:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}$
4:     $(J_1, J_2, \sigma) \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}; \omega)$          ▷ 1st run
5:     if $J_1 = 0$ or $J_2 = 0$ then
6:         return $(0, \epsilon, \epsilon, \epsilon, \epsilon)$
7:     end if
8:     pick $h'_{J_2}, \dots, h'_{\nu} \xleftarrow{\$} \mathsf{R}$
9:     set $\boldsymbol{h}' = (h_1, \dots, h_{J_2-1}, h'_{J_2}, \dots, h'_{\nu})$
10:    $(J_1', J_2', \sigma') \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}'; \omega)$          ▷ 2nd run
11:    if $\boxed{J_1 \neq J_1'}$ or $J_2 \neq J_2'$ or $h_{J_2} = h'_{J_2}$ then
12:        return $(0, \epsilon, \epsilon, \epsilon, \epsilon)$
13:    end if
14:    pick $h''_{J_1}, \dots, h''_{J_2-1}, h''_{J_2+1}, \dots, h''_{\nu} \xleftarrow{\$} \mathsf{R}$
15:    set $\boldsymbol{h}'' = (h_1, \dots, h_{J_1-1}, h''_{J_1}, \dots, h''_{J_2-1}, h_{J_2}, h''_{J_2+1}, \dots, h''_{\nu})$
16:    $(J_1'', J_2'', \sigma'') \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}''; \omega)$          ▷ 3rd run
17:    if $J_1' \neq J_1''$ or $J_2' \neq J_2''$ or $h_{J_1} = h''_{J_1}$ then
18:        return $(0, \epsilon, \epsilon, \epsilon, \epsilon)$
19:    end if
20:    pick $h'''_{J_2+1}, \dots, h'''_{\nu} \xleftarrow{\$} \mathsf{R}$
21:    set $\boldsymbol{h}''' = (h''_1, \dots, h''_{J_2-1}, h'_{J_2}, h'''_{J_2+1}, \dots, h'''_{\nu})$
22:    $(J_1''', J_2''', \sigma''') \longleftarrow \mathcal{B}(\mathsf{inst}, \boldsymbol{h}'''; \omega)$          ▷ 4th run
23:    if $\boxed{J_1'' \neq J_1'''}$ or $J_2'' \neq J_2'''$ then
24:        return $(0, \epsilon, \epsilon, \epsilon, \epsilon)$
25:    end if
26:    return $(1, \sigma, \sigma', \sigma'', \sigma''')$
27: end procedure
```

The success probability of $\mathsf{ExtF}_{\mathcal{B}}$ can be expressed in terms of the forking probability frk as defined in Corollary 3.14. In fact,

$$\begin{aligned}
\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) &= \mathsf{Pr}\left[b = 1 : \ \mathsf{inst} \xleftarrow{\$} \mathsf{Mq};\ (b, \sigma, \sigma', \sigma'', \sigma''') \longleftarrow \mathsf{ExtF}_{\mathcal{B}}(\mathsf{inst})\right] \\
&= \sum_{\substack{j_1, j_2 \in [\nu] \\ j_1 < j_2}} \mathsf{frk}(\mathcal{W}_{j_1, j_2}, j_1, j_2) \\
&= \mathsf{frk} \\
&\geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^3}{8 \cdot (\nu^2 - \nu)^3} - \frac{2}{|\mathsf{R}|} \right) \qquad \text{[by Corollary 3.14]} \\
&\geq \varepsilon \cdot \left(\frac{\varepsilon^3}{8 \cdot (\nu^2 - \nu)^3} - \frac{2}{|\mathsf{R}|} \right) \qquad \text{[by equation (17)]} \\
&\approx \frac{\varepsilon^4}{8 \cdot (\nu^2 - \nu)^3} \qquad (21)
\end{aligned}$$

where we refer to Lemma 3.13 for the definition of $\mathsf{frk}(\mathcal{W}_{j_1, j_2}, j_1, j_2)$.

MQ Solver. Our MQ problem solver is given in Algorithm 6. The solver Simu takes an MQ problem instance $(\mathcal{P}, \boldsymbol{y}^*)$ as input and runs $\mathsf{ExtF}_{\mathcal{B}}$. With probability $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$, the forking algorithm returns four signatures $\sigma, \sigma', \sigma''$ and σ''' for the same message. Using Proposition A.3, the probability of reaching step 19 of Algorithm 6 without abort is $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) \cdot \left(1 - (\frac{1}{2} - \frac{1}{2q})^r\right) \approx \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$ (by the choice of r involved in step 3 of Algorithm 4). If the underlying commitment scheme is computationally binding, then $\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^*$, where $\boldsymbol{x}^*$ is the output of the extractor at step 19. Therefore, the success probability of finding a valid solution is

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{MQ}}_{\mathsf{Simu}}(\kappa) &\geq \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) - \mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{Simu}}(\kappa) \\
&\geq \frac{\varepsilon^4}{8 \cdot (\nu^2 - \nu)^3} - \mathsf{Adv}^{\mathrm{Binding}}_{\mathsf{Simu}}(\kappa), \qquad (22)
\end{aligned}$$

Algorithm 6 MQ Solver

```
 1: procedure Simu($\mathcal{P}, \boldsymbol{y}^*$)        ▷ output $\boldsymbol{x}^* \in \mathbb{F}^n$ such that $\mathcal{P}(\boldsymbol{x}^*) = \boldsymbol{y}^*$
 2:     set inst = $(\mathcal{P}, \boldsymbol{y}^*)$
 3:     $(b, \sigma, \sigma', \sigma'', \sigma''') \longleftarrow \mathsf{ExtF}_{\mathcal{B}}(\mathsf{inst})$
 4:     if $b = 0$ then
 5:         abort
 6:     end if
 7:     parse $\sigma$ as $(\mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}_1, \mathbf{ch}_2, \mathbf{rs}_2)$
 8:     parse $\sigma'$ as $(\mathbf{ct}, \mathbf{ch}_1, \mathbf{rs}'_1, \mathbf{ch}'_2, \mathbf{rs}'_2)$        ▷ due to Eqn 18 - 19
 9:     parse $\sigma''$ as $(\mathbf{ct}, \mathbf{ch}'_1, \mathbf{rs}''_1, \mathbf{ch}_2, \mathbf{rs}''_2)$        ▷ due to Eqn 18 - 20
10:     parse $\sigma'''$ as $(\mathbf{ct}, \mathbf{ch}'_1, \mathbf{rs}'''_1, \mathbf{ch}'_2, \mathbf{rs}'''_2)$        ▷ due to Eqn 18 - 20
11:     find an $i \in [r]$ such that $\mathsf{ch}_{1,i} \neq \mathsf{ch}'_{1,i}$ and $\mathsf{ch}_{2,i} \neq \mathsf{ch}'_{2,i}$.
12:     if such $i$ is not found then
13:         abort
14:     end if
15:     set $\pi = (\mathsf{ct}_i, \mathsf{ch}_{1,i}, \mathsf{rs}_{1,i}, \mathsf{ch}_{2,i}, \mathsf{rs}_{2,i})$
16:     set $\pi' = (\mathsf{ct}_i, \mathsf{ch}_{1,i}, \mathsf{rs}_{1,i}, \mathsf{ch}'_{2,i}, \mathsf{rs}'_{2,i})$
17:     set $\pi'' = (\mathsf{ct}_i, \mathsf{ch}'_{1,i}, \mathsf{rs}'_{1,i}, \mathsf{ch}_{2,i}, \mathsf{rs}''_{2,i})$
18:     set $\pi''' = (\mathsf{ct}_i, \mathsf{ch}'_{1,i}, \mathsf{rs}'_{1,i}, \mathsf{ch}'_{2,i}, \mathsf{rs}'''_{2,i})$
19:     $\boldsymbol{x}^* \longleftarrow \mathsf{Extr}(\pi, \pi', \pi'', \pi''')$        ▷ Extr Such an extractor exists due to Lemma B.5
20:     return $\boldsymbol{x}^*$
21: end procedure
```

where in the last step, we use equation (21). □

Since the underlying commitment scheme is computational binding, one can approximate equation (22) as

$$\varepsilon \leq \nu^{3/2} \cdot \left(\mathsf{Adv}^{\mathrm{MQ}}_{\mathsf{Simu}}(\kappa)\right)^{1/4}. \tag{23}$$

4.3 Comparative Security of MQDSS Variants

Recall that the 5-pass and not the 3-pass variant of MQDSS was submitted for standardisation in the NIST PQC competition [NIS19]. In [CHR+16] it was argued that the signature length in the 5-pass variant is smaller compared to the 3-pass variant, as the number of parallel rounds required in the corresponding IDS would be lower due to the corresponding knowledge error. However, the previously mentioned attack [KZ20] based on concrete parameter choice of 5-pass MQDSS reveals the fallacy of this heuristic justification. In fact, a major implication of the attack in [KZ20] is that the number of rounds in the IDS needs to be increased for the 5-pass MQDSS. As a consequence, the 5-pass variant essentially loses its comparative advantage over 3-pass MQDSS.

It should be noted that the attack in [KZ20] works on concrete parameter choices without invalidating the asymptotic security guarantee provided in [CHR+16]. So, one may wonder what really went wrong! It is precisely what was missing from the original security proof. As we already pointed out, the reduction for 5-pass MQDSS in [CHR+16] was only qualitative in nature, without any quantitative bound on the probability of success of the MQDSS forger. It is specifically mentioned in [KZ20] that the attack on 5-pass MQDSS stems from the lack of tightness in security reduction. A similar problem has previously been identified for several other provably secure cryptographic schemes, as demonstrated in [CMS11, Zav12, CKM+16]. In other words, the attack [KZ20] could have been avoided if the actual choice of parameter for MQDSS had been guided by a concrete analysis of security reduction. However, no such bound was available before our result.

As is evident from our concrete analysis, the reductions for both 3 and 5-pass MQDSS are non-tight as there is a degradation in terms of the probability of success of the MQDSS forger compared to the MQ solver (see equations (23) and (16)). However, a quick comparison of the two equations is enough to conclude that the degradation is far more severe in the 5-pass variant than in the 3-pass variant. In other words, there could be some potential low-probability attacks that are ruled out by the security reduction for the 3-pass MQDSS but not for the 5-pass. The attack proposed in [KZ20], which works on 5-pass but not 3-pass MQDSS, seems to be a concrete example of this scenario. In that sense, the 3-pass MQDSS provides a better concrete security assurance. However, the reduction for the 3-pass variant still being far from tight, some potential attacks cannot be ruled out unless the concrete parameter choice is guided by the analysis provided in this paper. So, further cryptanalysis of MQDSS on this line or giving a reduction that is tighter than ours to get a better concrete security assurance are some of the interesting open questions arising from our work.

5 On the Security of Multivariate Blind Signature Scheme

Petzoldt et al. [PSM17] introduced a multivariate-based blind signature scheme which we refer to as MBSS. This scheme is based on the 5-pass IDS of Sakumoto et al. [SSH11] and the two-layered Rainbow [DS05]. They claimed that the construction achieves universal-one-more-unforgeability (UOMF) (c.f., Definition 2.7) under the intractability of the so-called PR-inversion problem[7] in the random oracle model. However, the security argument provided in [PSM17] is rather sketchy with several critical gaps [Maj21]. Furthermore, UOMF is a significantly weaker security model than the standard one, namely, one-more unforgeability (OMF) (cf. Definition 2.8). Therefore, it is crucial to investigate whether MBSS realises OMF security which is the focus of this section.

Note that Rainbow signature was introduced to improve efficiency upon the UOV signature. Due to the recent attack by Beullens [Beu22] on Rainbow, the UOV signature gets attention again from the crypto community. With a judicious choice of parameters, UOV withstands all the attacks available so far.

We reproduce the MBSS in Section 5.1 with slight changes from the original [PSM17] as follows. (i) The 5-pass IDS is replaced by the 3-pass IDS [SSH11] based on the results of the previous section. (ii). Rainbow is replaced with the UOV signature due to the attack mentioned earlier. Note that in construction, UOV is used in a modular way, particularly as an MQ-based trapdoor function. The security reduction maintains the same framework in which the public maps of the underlying UOV signature scheme (and not the secret key) and commitment polynomial maps are utilised. Hence, Rainbow could be easily replaced with any of the secure variants of UOV, such as MAYO, PROV, and TUOV. (iii) In addition, the original commitment function $\mathcal{R} : \mathbb{F}^m \to \mathbb{F}^m$ is replaced with $\mathcal{R} : \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$, where $\bar{n} \ll m$, as suggested by Beullens [Beu24] (see Remark 5.2). (iv) In addition, there are some minor modifications, namely the challenge computation involved in the underlying MQDSS (see Footnote 8).

In Section 5.2, we show that the 3-pass MBSS is OMF secure in the random oracle model. We note that a similar strategy can be applied to the MBSS based on the 5-pass MQDSS, though it will be a bit complicated due to the involvement of the 5-pass IDS.

5.1 Multivariate Blind Signature Scheme

Here we describe the multivariate blind signature scheme from [PSM17], using the 3-pass IDS of [SSH11] (see Figure 4). There are two hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ in the construction: $\mathcal{H}_1$ is involved in the underlying UOV signature and $\mathcal{H}_2$ is applied in the context of the Fiat-Shamir transform [FS86]. We consider the input and output of the hash functions in a nested manner. That is, the output of $\mathcal{H}_1$ will be a part of the input of $\mathcal{H}_2$. Formally, the construction is as follows.

$\mathsf{KeyGen}(\kappa)$. Let (n, m, q) be the set of UOV parameters defined by κ with $m = o$ and $n = v + o$. Choose $\bar{n}$ to be much smaller than m (see Remark 5.2). Let $\mathcal{M}$ be the message space. Choose a number $r \in \mathbb{N}$

[7] Given $(\mathcal{P}, \mathcal{R}) \in \mathcal{P}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m) \times \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m)$, find $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*) \in \mathbb{F}^n \times \mathbb{F}^{\bar{n}}$ such that $\mathcal{P}(\boldsymbol{x}_1^*) + \mathcal{R}(\boldsymbol{x}_2^*) = \mathbf{0}$. In other words, it is a special case of the PR-STI problem, where the target $\boldsymbol{y}$ is set to $\mathbf{0}$.

such that $(2/3)^r = \mathsf{negl}(\kappa)$. This choice essentially makes the knowledge error of IDS^r negligible thanks to Lemma B.3. Let $\mathbb{F}$ be a field of size q. Let $\mathcal{H}_1 : \mathcal{M} \to \mathbb{F}^m$ and $\mathcal{H}_2 : \{0,1\}^* \to \{0,1,2\}^r$ be cryptographic hash functions. Let $\mathcal{C} = (\mathsf{CSetup}, \mathsf{Commit}, \mathsf{Open})$ be an efficient commitment scheme. The steps of the key-generation algorithm are given below.

1. pick $\mathcal{F} \xleftarrow{\$} \mathcal{F}_{\mathsf{uov}}(\mathbb{F}^n, \mathbb{F}^m)$ and $\mathcal{T} \xleftarrow{\$} \mathsf{invAff}(\mathbb{F}^n, \mathbb{F}^n)$
2. set $\mathcal{P} = \mathcal{F} \circ \mathcal{T}$ as the UOV public map
3. pick $\mathcal{R} \xleftarrow{\$} \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m)$
4. set $\Gamma = (\mathcal{P}, \mathcal{R})$
5. run $\mathsf{ck} \longleftarrow \mathsf{CSetup}(\kappa)$
6. set $\mathsf{pk} = (\Gamma, \mathcal{H}_1, \mathcal{H}_2, \mathsf{ck}, r)$ and $\mathsf{sk} = (\mathcal{S}, \mathcal{F}, \mathcal{T}, \mathcal{H}_1, \mathcal{H}_2, \mathsf{ck}, r)$
7. return $(\mathsf{pk}, \mathsf{sk})$

Sign. This is an interactive protocol between a user U and a signer S. The inputs to U and S are $(\mathsf{pk}, \mathrm{M})$ and $(\mathsf{pk}, \mathsf{sk})$, respectively. It consists of two stages: the first is interactive, and the second is non-interactive. In the former case U gets a blind signature from S on a message of her choice, and the latter is for the generation of the corresponding unblinded signature.

Interactive Stage. The interactive steps between S and U are given below.

1. U picks $\bar{\boldsymbol{z}} \xleftarrow{\$} \mathbb{F}^{\bar{n}}$ and computes $\boldsymbol{w} = \mathcal{H}_1(\mathrm{M})$
2. U computes $\widetilde{\boldsymbol{w}} = \boldsymbol{w} - \mathcal{R}(\bar{\boldsymbol{z}})$ and sends $\widetilde{\boldsymbol{w}}$ to S
3. S then runs $\boldsymbol{z} \longleftarrow \mathsf{UOVInv}(\mathsf{sk}^*, \widetilde{\boldsymbol{w}})$, where $\mathsf{sk}^* = (\mathcal{S}, \mathcal{F}, \mathcal{T})$ and sends $\widetilde{\sigma} = \boldsymbol{z}$ back to U
4. U checks if $\Gamma(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) \overset{?}{=} \boldsymbol{w}$, otherwise aborts
5. U then sets the witness to $\boldsymbol{\gamma} = (\boldsymbol{z}, \bar{\boldsymbol{z}})$ for the statement $(\Gamma, \boldsymbol{w})$, i.e., $\Gamma(\boldsymbol{\gamma}) = \boldsymbol{w}$

We refer to $\widetilde{\boldsymbol{w}}$ and $\widetilde{\sigma}$ as blind message and signature, respectively.

Non-Interactive Stage. Through this stage U generates a proof that it knows a witness to the statement $(\Gamma, \boldsymbol{w})$. The proof is essentially constructed using the 3-pass IDS of Sakumoto et al. [SSH11] followed by the Fiat-Shamir transform [FS86] as described in the following.

1. let G be the polar form of system $\Gamma : \mathbb{F}^n \times \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$
2. pick $\boldsymbol{a}_{0,i}, \boldsymbol{b}_{0,i} \xleftarrow{\$} \mathbb{F}^{n+\bar{n}}$ and $\boldsymbol{c}_{0,i} \xleftarrow{\$} \mathbb{F}^m$ for $i \in [r]$
3. set $\boldsymbol{a}_{1,i} = \boldsymbol{\gamma} - \boldsymbol{a}_{0,i}$, $\boldsymbol{b}_{1,i} = \boldsymbol{a}_{0,i} - \boldsymbol{b}_{0,i}$ and $\boldsymbol{c}_{1,i} = \Gamma(\boldsymbol{a}_{0,i}) - \boldsymbol{c}_{0,i}$ for $i \in [r]$
4. for each $i \in [r]$, compute the following.
   (a) $\mathsf{ct}_{0,i} \longleftarrow \mathsf{Commit}(\boldsymbol{a}_{1,i}, \mathsf{G}(\boldsymbol{b}_{0,i}, \boldsymbol{a}_{1,i}) + \boldsymbol{c}_{0,i})$
   (b) $\mathsf{ct}_{1,i} \longleftarrow \mathsf{Commit}(\boldsymbol{b}_{0,i}, \boldsymbol{c}_{0,i})$
   (c) $\mathsf{ct}_{2,i} \longleftarrow \mathsf{Commit}(\boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$
5. set $\mathbf{ct} = (\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \ldots, \mathsf{ct}_{0,r}, \mathsf{ct}_{1,r}, \mathsf{ct}_{2,r})$ and compute $\mathbf{ch} = \mathcal{H}_2(\boldsymbol{w}, \mathbf{ct})$[8]
6. parse $\mathbf{ch}$ as $(\mathsf{ch}_1, \ldots, \mathsf{ch}_r)$

[8]The original paper [PSM17] did not explain the challenge computations involved in the 5-pass MQDSS. It just mentioned that $\mathbf{ch}_1$ (first challenge) is derived from $(\mathcal{D}, \mathbf{ct})$, where $\mathcal{D} = \mathcal{H}(\mathcal{C}, \boldsymbol{w})$, $\mathcal{C} = \mathcal{H}(\mathcal{P}, \boldsymbol{w})$ and $\boldsymbol{w} = \mathcal{H}(\mathrm{M})$. That is, the first argument $\mathcal{D}$ involved in the derivation of $\mathbf{ch}_1$ is computed through three levels of hashing. Similarly, $\mathbf{ch}_2$ is derived. The paper did not give any justification for such hashing. In contrast, we consider only one level of hashing.

7. for each $i \in [r]$, do the following.
   (a) if $\mathsf{ch}_i = 0$, set $\mathsf{rs}_i = (\boldsymbol{a}_{0,i}, \boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$
   (b) if $\mathsf{ch}_i = 1$, set $\mathsf{rs}_i = (\boldsymbol{a}_{1,i}, \boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$
   (c) if $\mathsf{ch}_i = 2$, set $\mathsf{rs}_i = (\boldsymbol{a}_{1,i}, \boldsymbol{b}_{0,i}, \boldsymbol{c}_{0,i})$
8. set $\mathbf{rs} = (\mathsf{rs}_1, \dots, \mathsf{rs}_r)$
9. return unblind signature $\sigma = (\mathbf{ct}, \mathbf{rs})$ on M

$\mathsf{Ver}(\mathsf{pk}, \mathrm{M}, \sigma)$. It consists of the following steps.

1. parse σ as $(\mathbf{ct}, \mathbf{rs})$, where $\mathbf{ct} = (\mathsf{ct}_{0,1}, \mathsf{ct}_{1,1}, \mathsf{ct}_{2,1}, \dots, \mathsf{ct}_{0,r}, \mathsf{ct}_{1,r}, \mathsf{ct}_{2,r})$ and $\mathbf{rs} = (\mathsf{rs}_1, \dots, \mathsf{rs}_r)$
2. compute $\boldsymbol{w} = \mathcal{H}_1(\mathrm{M})$ and $\mathbf{ch} = \mathcal{H}_2(\boldsymbol{w}, \mathbf{ct})$
3. for each $i \in [r]$, do the following:
   (a) if $\mathsf{ch}_i = 0$, parse rs_i as $(\boldsymbol{a}_{0,i}, \boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$ and check if $\mathsf{ct}_{1,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_{0,i} - \boldsymbol{b}_{1,i}, \Gamma(\boldsymbol{a}_{0,i}) - \boldsymbol{c}_{1,i})$ and $\mathsf{ct}_{2,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$
   (b) if $\mathsf{ch}_i = 1$, parse rs_i as $(\boldsymbol{a}_{1,i}, \boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$ and check if $\mathsf{ct}_{0,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_{1,i}, \boldsymbol{w} - \Gamma(\boldsymbol{a}_{1,i}) - \mathsf{G}(\boldsymbol{b}_{1,i}, \boldsymbol{a}_{1,i}) - \boldsymbol{c}_{1,i} + \Gamma(\mathbf{0}))$ and $\mathsf{ct}_{2,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_{1,i}, \boldsymbol{c}_{1,i})$
   (c) if $\mathsf{ch}_i = 2$, parse rs_i as $(\boldsymbol{a}_{1,i}, \boldsymbol{b}_{0,i}, \boldsymbol{c}_{0,i})$ and check if $\mathsf{ct}_{0,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_{1,i}, \mathsf{G}(\boldsymbol{b}_{0,i}, \boldsymbol{a}_{1,i}) + \boldsymbol{c}_{0,i})$ and $\mathsf{ct}_{1,i} \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_{0,i}, \boldsymbol{c}_{0,i})$
   (d) return 0 if any of the above checks fails
4. return 1

Correctness: For all keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and all messages $\mathrm{M} \in \mathcal{M}$, we have to show that an unblinded signature σ on M generated using $(\mathsf{pk}, \mathsf{sk})$ is accepted by the verification algorithm. Note that here σ is essentially a proof that ensures that the user knows a witness $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ such that $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \mathcal{H}_1(\mathrm{M})$. Since the proof is generated using the 3-pass MQDSS, the correctness of the MBSS follows from that of the 3-pass MQDSS.

Remark 5.1. Note that $\mathbf{rs}$ implicitly contains the randomnesses involved in Commit of the non-interactive stage of the sign algorithm and the same will be used in step 3 of the verification algorithm. We sometimes express the signature as $\sigma = \big(\{\mathsf{ct}_i, \mathsf{ch}_i, \mathsf{rs}_i\}_{i\in[r]}\big)$, where $\mathbf{ch} = (\mathsf{ch}_1, \dots, \mathsf{ch}_r) = \mathcal{H}_2(\boldsymbol{w}, \mathbf{ct})$ and $\boldsymbol{w} = \mathcal{H}_1(\mathrm{M})$.

Remark 5.2. Recently, Beullens [Beu24] showed that a collision for $\mathcal{R} : \mathbb{F}^{\bar{n}} \to \mathbb{F}^m$ is efficiently computable when $\bar{n} \geq m$, and therefore, the original MBSS [PSM17] is not OMF secure. Hence, we take $\bar{n}$ to be much smaller than m as a countermeasure recommended by Beullens to resist his attack.

5.2 A Concrete Security Treatment of MBSS

In this section, we provide a proof of OMF security for MBSS from the PR-mSTI and CMQ problems. Let $(\mathrm{M}_1, \sigma_1), \dots, (\mathrm{M}_{\ell+1}, \sigma_{\ell+1})$ be the unblinded message-signature pairs produced by an attacker $\mathcal{A}$ after making ℓ many blind signature queries. For $i \in [\ell+1]$, let $\boldsymbol{w}_i = \mathcal{H}_1(\mathrm{M}_i)$ and let J_i denote the index of $\mathcal{H}_2$-oracle for the query argument $(\boldsymbol{w}_i, \mathbf{ct}^{(i)})$. Let $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1), \dots, (\boldsymbol{z}_{\ell+1}, \bar{\boldsymbol{z}}_{\ell+1})$ be the witnesses involved in those $\ell+1$ message-signature pairs. In fact, if we rewind $\mathcal{A}$ at index J_i twice, we can extract the witness $(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i)$. We expect $(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i)$ to be a solution of the PR-mSTI problem for some $i \in [\ell+1]$ if we appropriately programme the random oracles. However, we first describe a scenario in which one fails to extract the solution of the PR-mSTI problem. Suppose for the first ℓ signatures, $\mathcal{A}$ just utilises the $\mathcal{O}_{\mathsf{BS}}$-oracle. That is, for each $i \in [\ell]$, $\mathcal{A}$ first computes/samples $\bar{\boldsymbol{z}}_i$ and then makes query on $\widetilde{\boldsymbol{w}}_i = \boldsymbol{w}_i - \mathcal{R}(\bar{\boldsymbol{z}}_i)$ to $\mathcal{O}_{\mathsf{BS}}$-oracle and let $\boldsymbol{z}_i$ be the corresponding reply. Then, $\mathcal{A}$ easily derive the signature σ_i for the message M_i using $(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i)$ as a witness. If we programme $\mathcal{H}_1(\mathrm{M}_i)$ by the challenge $\boldsymbol{w}^*$ involved in the PR-mSTI problem instance for any $i \in [\ell]$ and extract out the witness $(\boldsymbol{z}_i, \bar{\boldsymbol{z}}_i)$, still we do not get a valid solution of the PR-mSTI problem. This is simply because the solution is trivial and therefore ruled out by the second requirement (2) of the problem (see Definition 2.19). On the other hand, one may expect that the signature $\sigma_{\ell+1}$ for the extra message $\mathrm{M}_{\ell+1}$ would be the forgery, because $\mathcal{A}$ cannot utilise the $\mathcal{O}_{\mathsf{BS}}$-oracle any more. However, the reduction may still fail if the value $\widetilde{\boldsymbol{w}}_{\ell+1} = \boldsymbol{w}_{\ell+1} - \mathcal{R}(\bar{\boldsymbol{z}}_{\ell+1})$ appears as an argument of the $\mathcal{O}_{\mathsf{BS}}$-oracle, that is, $\widetilde{\boldsymbol{w}}_{\ell+1} \in \{\widetilde{\boldsymbol{w}}_1, \ldots, \widetilde{\boldsymbol{w}}_\ell\}$. Then, this again violates the second requirement of the PR-mSTI problem. The above scenario is captured by what we call a collision set.

Definition 5.1 (Collision Set and Collision-Free Set)**.** Suppose an attacker produces $(\ell + 1)$ many distinct message-signature pairs $(\mathrm{M}_1, \sigma_1), \ldots, (\mathrm{M}_{\ell+1}, \sigma_{\ell+1})$ after making ℓ many blind signature queries. Let $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1), \ldots, (\boldsymbol{z}_{\ell+1}, \bar{\boldsymbol{z}}_{\ell+1})$ be the witnesses involved in $\ell + 1$ forgeries. Then, the above forgeries are said to form a *collision set* if there exist $i, j \in [\ell + 1]$ with $i \neq j$ such that $\widetilde{\boldsymbol{w}}_i = \widetilde{\boldsymbol{w}}_j$, where $\boldsymbol{w}_i = \mathcal{H}_1(\mathrm{M}_i)$, $\boldsymbol{w}_j = \mathcal{H}_1(\mathrm{M}_j)$, $\widetilde{\boldsymbol{w}}_i = \boldsymbol{w}_i - \mathcal{R}(\bar{\boldsymbol{z}}_i)$ and $\widetilde{\boldsymbol{w}}_j = \boldsymbol{w}_j - \mathcal{R}(\bar{\boldsymbol{z}}_j)$. When the above forgeries do not form a collision set, the forgeries are said to form a *collision-free set.*

Next, we define the notion of non-trivial forgery.

Definition 5.2 (Non-trivial Forgery)**.** Suppose an attacker produces a set of $(\ell+1)$ many message-signature pairs $\mathfrak{S} = \{(\mathrm{M}_1, \sigma_1), \ldots, (\mathrm{M}_{\ell+1}, \sigma_{\ell+1})\}$ after making ℓ many blind signature queries. Let $\mathsf{arg}_1, \ldots, \mathsf{arg}_\ell$ be queries to the $\mathcal{O}_{\mathsf{BS}}$-oracle. Let $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1), \ldots, (\boldsymbol{z}_{\ell+1}, \bar{\boldsymbol{z}}_{\ell+1})$ be the witnesses involved in $\ell + 1$ forgeries. Then, the set $\mathfrak{S}$ is said to contain a *non-trivial forgery* if there exists some $i \in [\ell+1]$ such that $\widetilde{\boldsymbol{w}}_i \notin \{\mathsf{arg}_1, \ldots, \mathsf{arg}_\ell\}$, where $\boldsymbol{w}_i = \mathcal{H}_1(\mathrm{M}_i)$ and $\widetilde{\boldsymbol{w}}_i = \boldsymbol{w}_i - \mathcal{R}(\bar{\boldsymbol{z}}_i)$.

Note that a collision-free set always contains a non-trivial forgery. In fact, we have two lists of distinct values $\{\mathsf{arg}_1, \ldots, \mathsf{arg}_\ell\}$ and $\{\widetilde{\boldsymbol{w}}_1, \ldots, \widetilde{\boldsymbol{w}}_{\ell+1}\}$. Since the second list contains more elements than the first, there exists an $i \in [\ell]$ such that $\widetilde{\boldsymbol{w}}_i \notin \{\mathsf{arg}_1, \ldots, \mathsf{arg}_\ell\}$. Therefore, we can say that the set $\mathfrak{S}$ returned by $\mathcal{A}$ either forms a collision set or contains a non-trivial forgery. When there is a non-trivial forgery, we construct a solver for the PR-mSTI problem, otherwise a solver for the CMQ problem. For solving those problems, we respectively use Corollaries 3.10 and 3.12, which are based on the 2-single-splitting-forking and the (2,2)-multi-splitting-forking.

Theorem 5.1. *If the PR-mSTI problem (cf. Definition 2.19) and the CMQ problem (c.f., Definition 2.16) are intractable and the underlying commitment scheme is computationally binding (c.f., Definition 2.3), then the MBSS described in Section 5.1 is OMF-secure (c.f., Definition 2.8) in the random oracle model.*

Proof. Let $\mathcal{A}$ be an adversary who can break OMF-security of the MBSS. Then, we break either the CMQ-problem or the PR-mSTI problem using $\mathcal{A}$ as a subroutine. In the reduction, $\mathcal{A}$ makes queries to random oracles and $\mathcal{O}_{\mathsf{BS}}$-oracle and, at the end, produces a set $\mathfrak{S}$ of $(\ell + 1)$ message-signature pairs, where ℓ is the number of queries to $\mathcal{O}_{\mathsf{BS}}$-oracle. We claim that $\mathfrak{S}$ is a collision-free set (cf. Definition 5.1). Suppose not, then using Lemma 5.2 one can create a solver for breaking the CMQ problem. We, therefore, assume that $\mathfrak{S}$ contains at least one nontrivial forgery (cf. Definition 5.2). In the rest of the proof, we create a solver for the PR-mSTI problem.

Recall that there are two hash functions $\mathcal{H}_1 : \mathcal{M} \to \mathbb{F}^m$ and $\mathcal{H}_2 : \{0,1\}^* \to \{0,1,2\}^r$. We treat them as random oracles. Since $\mathcal{H}_1$ and $\mathcal{H}_2$ are finite sets, they can be represented by some enumerations. Let $\mathsf{R}_1 = \{1, \ldots, q^m\}$ and $\mathsf{R}_2 = \{1, \ldots, 3^r\}$ be the enumerations of $\mathbb{F}^m$ and $\{0,1,2\}^r$, respectively. Then, the elements of $\mathbb{F}^m$ and $\{0,1,2\}^r$ are uniquely and efficiently determined by their indices in the enumerations

R_1 and R_2, respectively. In fact, if $\mathsf{Enu}_1 : \mathsf{R}_1 \to \mathbb{F}^m$ and $\mathsf{Enu}_2 : \mathsf{R}_2 \to \{0,1,2\}^r$ are two one-to-one functions, then the i-th elements in $\mathbb{F}^m$ and $\{0,1,2\}^r$ are represented by $\mathsf{Enu}_1(i)$ and $\mathsf{Enu}_2(i)$, respectively. W.l.o.g, we assume that such functions exist and are efficiently computable.

Let the number of queries to the $\mathcal{H}_1$ and $\mathcal{H}_2$-oracles be ν_1 and ν_2 respectively. Let $\nu = \nu_1 + \nu_2$. So, the upper bound on the total number of queries is $\widetilde{\nu} = \nu + \ell$. Let us define two index sets $\mathsf{R} = \{1, 2, \ldots, \varphi\}$ and $\widetilde{\mathsf{R}} = \{1, 2, \ldots, \widetilde{\varphi}\}$, where $\varphi = 3^r q^m$, $\widetilde{\varphi} = |\mathbb{F}^v|^t \cdot \varphi = q^{t\cdot v} \cdot \varphi$, $q = |\mathbb{F}|$, t is the expected number of runs of the loop in the UOV signing algorithm and v is the number of vinegar variables. Note that $(\mathbb{F}^v)^t$ is an enumerable set. Similarly as above, we can find an efficient computable function $\mathsf{Enu}_3 : \mathsf{R}_3 \to (\mathbb{F}^v)^t$ such that the i-th element of $(\mathbb{F}^v)^t$ is described by $\mathsf{Enu}_3(i)$, where $\mathsf{R}_3 = \{1, \ldots, q^{t\cdot v}\}$.

Next, we define two first-level encoding functions $\mathsf{Enc}_{\mathsf{RO}} : \widetilde{\mathsf{R}} \to \mathsf{R}$ and $\mathsf{Enc}_{\mathsf{BS}} : \widetilde{\mathsf{R}} \to \mathsf{R}_3$ as follows.

$$\mathsf{Enc}_{\mathsf{RO}}(x) = \left\lceil \frac{x}{q^{t\cdot v}} \right\rceil \text{ and } \mathsf{Enc}_{\mathsf{BS}}(x) = \left\lceil \frac{x}{\varphi} \right\rceil \text{ for } x \in \widetilde{\mathsf{R}}.$$

These encoding functions are utilized for answering the random oracle queries and sampling the randomness involved in the blind signature generation respectively. For any $y \in \mathsf{R}$, the preimage set of y is given by

$$\mathsf{Enc}_{\mathsf{RO}}^{-1}(y) = \{q^{t\cdot v} \cdot (y-1) + 1, q^{t\cdot v} \cdot (y-1) + 2, \ldots, q^{t\cdot v} \cdot y\}.$$

Note that one can efficiently sample a random element from $\mathsf{Enc}_{\mathsf{RO}}^{-1}(y)$. We now define two second-level encoding functions $\mathsf{Enc}_1 : \mathsf{R} \to \mathsf{R}_1$ and $\mathsf{Enc}_2 : \mathsf{R} \to \mathsf{R}_2$ as follows.

$$\mathsf{Enc}_1(x) = \left\lceil \frac{x}{3^r} \right\rceil \text{ and } \mathsf{Enc}_2(x) = \left\lceil \frac{x}{q^m} \right\rceil \text{ for } x \in \mathsf{R}.$$

For any $y \in \mathsf{R}_1$, the preimage set of y is defined by

$$\mathsf{Enc}_1^{-1}(y) = \{3^r \cdot (y-1) + 1, 3^r \cdot (y-1) + 2, \ldots, 3^r \cdot y\}.$$

Similarly as above, one can efficiently sample a uniform element from $\mathsf{Enc}_1^{-1}(y)$.

To properly answer the queries made by $\mathcal{A}$, we use the following properties related to the above defined encoding functions.

1. $\mathsf{Enc}_{\mathsf{RO}} : \widetilde{\mathsf{R}} \to \mathsf{R}$, $\mathsf{Enc}_{\mathsf{BS}} : \widetilde{\mathsf{R}} \to \mathsf{R}_3$, $\mathsf{Enc}_1 : \mathsf{R} \to \mathsf{R}_1$ and $\mathsf{Enc}_2 : \mathsf{R} \to \mathsf{R}_2$ are regular.
2. $\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}} : \widetilde{\mathsf{R}} \to \mathbb{F}^m$, $\mathsf{Enu}_2 \circ \mathsf{Enc}_2 \circ \mathsf{Enc}_{\mathsf{RO}} : \widetilde{\mathsf{R}} \to \{0,1,2\}^r$ and $\mathsf{Enu}_3 \circ \mathsf{Enc}_{\mathsf{BS}} : \widetilde{\mathsf{R}} \to (\mathbb{F}^v)^t$ are regular functions due to property (1).
3. All these functions are efficiently computable.
4. When $u \xleftarrow{\$} \widetilde{\mathsf{R}}$, then $(\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}})(u)$, $(\mathsf{Enu}_2 \circ \mathsf{Enc}_2 \circ \mathsf{Enc}_{\mathsf{RO}})(u)$ and $(\mathsf{Enu}_3 \circ \mathsf{Enc}_{\mathsf{BS}})(u)$ are uniformly distributed over $\mathbb{F}^m$, $\{0,1,2\}^r$ and $(\mathbb{F}^v)^t$, respectively due to property (2).
5. For any $\boldsymbol{z} \in \mathbb{F}^m$, one can efficiently sample a uniform element from $(\mathsf{Enu}_1 \circ \mathsf{Enc}_1)^{-1}(\boldsymbol{z})$. In fact, let $y = \mathsf{Enu}_1^{-1}(\boldsymbol{z}) \in \mathsf{R}_1$. Then sample $x \xleftarrow{\$} \mathsf{Enc}_1^{-1}(y)$.
6. When $\boldsymbol{z} \xleftarrow{\$} \mathbb{F}^m$ and $x \xleftarrow{\$} (\mathsf{Enu}_1 \circ \mathsf{Enc}_1)^{-1}(\boldsymbol{z})$, then x is uniform over R as the function $\mathsf{Enu}_1 \circ \mathsf{Enc}_1 : \mathsf{R} \to \mathbb{F}^m$ is regular.

For better readability, the connectivity of $\mathbb{F}^m$, $\{0,1,2\}^r$, and $(\mathbb{F}^v)^t$ respectively from $\widetilde{\mathsf{R}}$ through R_1, R_2, and R_3 is shown in the following diagram.

$$\begin{tikzcd}
 & & & & \mathsf{R}_1 \arrow[r, "\mathsf{Enu}_1"] & \mathbb{F}^m \\
(\mathbb{F}^v)^t & \mathsf{R}_3 \arrow[l, "\mathsf{Enu}_3"'] & \widetilde{\mathsf{R}} \arrow[l, "\mathsf{Enc}_{\mathsf{BS}}"'] \arrow[r, "\mathsf{Enc}_{\mathsf{RO}}"] & \mathsf{R} \arrow[ur, "\mathsf{Enc}_1"] \arrow[dr, "\mathsf{Enc}_2"'] & & \\
 & & & & \mathsf{R}_2 \arrow[r, "\mathsf{Enu}_2"'] & \{0,1,2\}^r
\end{tikzcd}$$

Note that in some of the intermediate games of our hybrid argument, the secret key is included in the PR-mSTI problem instances. We denote this problem by $\overline{\mathsf{Prmsti}}$ with $\overline{\mathsf{Prmsti}} = \overline{\mathsf{Prmsti}}_1 \times \mathsf{Prmsti}_2$, where $\overline{\mathsf{Prmsti}}_1$ is the set consisting of all tuples of the form $((\mathcal{P}, \mathcal{R}), \mathsf{sk}^*)$ with $(\mathcal{P}, \mathcal{R}) \in \mathsf{Prmsti}_1$ and sk^* is the secret key associated with the UOV public key $\mathcal{P}$. Of course, such a problem will be trivially easy and our security reduction does not rely on such problem instances. Such easy instances are used to simplify the otherwise involved security argument. In fact, we consider a series of forking algorithms to solve the PR-mSTI problem. Using a hybrid argument, we show that the forking probability of all these algorithms are essentially the same. Our initial version of the forking algorithms (Algorithm 8) takes input from $\overline{\mathsf{Prmsti}}$, whereas the final version (Algorithm 11) takes input from Prmsti (the actual PR-mSTI problem).

Wrapper Algorithm. For $k \in [\ell+1]$, we construct a wrapper algorithm $\mathcal{B}_k$ (see Algorithm 7) that takes $(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}) \in \overline{\mathsf{Prmsti}}_1 \times \widetilde{\mathsf{R}}^{\widetilde{\nu}}$ as input and creates an environment for $\mathcal{A}$. Throughout our analysis, $1/|\widetilde{\mathsf{R}}| = \mathsf{negl}(\kappa)$ due to the choice of r. When $\mathcal{A}$ returns the message-signature pairs $(\ell+1)$, $\mathcal{B}_k$ returns the pair (J, σ). Here σ is called side output and $J \in [0, \widetilde{\nu}]$ is called the *forking index*, where the message M and the commitment $\mathbf{ct}$ associated with σ are committed by $\mathcal{H}_2$-hash. Note that the random oracle and blind signature queries made by $\mathcal{A}$ may appear in any order. The wrapper $\mathcal{B}_k$ handles random oracle queries by evaluating appropriate encoding functions on ν many points of $\widetilde{\boldsymbol{h}}$ (see the first column of Figure 3). The queries to the $\mathcal{O}_{\mathsf{BS}}$-oracle are handled using sk (part of $\overline{\mathsf{inst}}_1$) and the random coins synthesised by evaluating suitable encoding functions on ℓ many points of $\widetilde{\boldsymbol{h}}$ (see the 2nd column of Figure 3). All queries of $\mathcal{A}$ are perfectly handled by $\mathcal{B}_k$ due to the property (4) mentioned above. Note that for the k-th forgery, except the negligible probability of guessing the correct output from $\{0,1,2\}^r$, $\mathcal{A}$ makes a query $(\boldsymbol{w}_k, \mathbf{ct}^{(k)})$ to $\mathcal{H}_2$ oracle. $\mathcal{B}_k$ can check it by searching for the tuple $(*, (\boldsymbol{w}_k, \mathbf{ct}^{(k)}), *) \in \mathsf{List}$. When $\mathcal{A}$ makes such a query, then the corresponding index (that is, the first entry in the tuple) will be denoted by J_k. Otherwise, we say that J_k is not defined. Furthermore, note that J_k is basically a random variable that depends on $(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}; \omega)$, where $\omega \in \Omega$ is the internal random coin used by $\mathcal{A}$ (which is supplied by $\mathcal{B}_k$).

Algorithm 7 Wrapper for PR-mSTI

```
 1: procedure $\mathcal{B}_k(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}})$    ▷ compute a signature $\sigma_k$ and the corresponding forking-index $J_k$
 2:     run $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(\kappa)$
 3:     pick $r \in \mathbb{N}$ such that $(2/3)^r = \mathsf{negl}(\kappa)$
 4:     $\mathsf{List} \leftarrow \emptyset$ and $\mathsf{ctr} \leftarrow 0$    ▷ List stores tuples of the form $(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))$ for $i \in [2]$
 5:     $\left\{(\mathrm{M}_i, (\mathbf{ct}^{(i)}, \mathbf{rs}^{(i)}))_{i\in[\ell+1]}\right\} \longleftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{all}}}(\overline{\mathsf{inst}}_1, \mathsf{ck}, r)$    ▷ here $\mathcal{O}_{\mathsf{all}} = \{\mathcal{O}_{\mathcal{H}_1}, \mathcal{O}_{\mathcal{H}_2}, \mathcal{O}_{\mathsf{BS}}\}$ and $\mathcal{O}_{\mathcal{H}_i}$ for $i = 1, 2$ are defined in Figure 3
 6:     if $J_k$ is not defined then
 7:         return $(0, \epsilon)$
 8:     end if
 9:     for $i = 1$ to $\ell + 1$ do
10:         $\boldsymbol{w}_i \xleftarrow{Q} \mathcal{H}_1(\mathrm{M}_i)$
11:         $\mathbf{ch}^{(i)} \xleftarrow{Q} \mathcal{H}_2(\boldsymbol{w}_i, \mathbf{ct}^{(i)})$
12:         set $\sigma_i = (\mathbf{ct}^{(i)}, \mathbf{ch}^{(i)}, \mathbf{rs}^{(i)})$
13:     end for
14:     if $\mathsf{Ver}(\mathsf{pk}, \mathrm{M}_i, \sigma_i) = 0$ for some $i \in [\ell]$ or $\mathrm{M}_i = \mathrm{M}_j$ for some $i, j \in [\ell+1]$ with $i \neq j$ then    ▷ where $\mathsf{pk} = (\overline{\mathsf{inst}}_1, \mathsf{ck})$
15:         return $(0, \epsilon)$
16:     end if
17:     set $J = J_k$ and $\sigma = \sigma_k$
18:     return $(J, \sigma)$
19: end procedure
```

We will define the success probability of $\mathcal{B}_k$, denoted by acc, following the style of Corollary 3.10. Let Ω be the domain from which $\mathcal{A}$ gets its randomness. Recall that $\mathcal{W} = \{(\overline{\mathsf{inst}}_1, \omega, \widetilde{\boldsymbol{h}}) \in \overline{\mathsf{Prmsti}}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}} :$

```
$\mathcal{O}_{\mathcal{H}_i}(\mathsf{arg})$: // for $i = 1, 2$
1: if $\mathcal{H}_i(\mathsf{arg})$ is not defined then
2:     $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
3:     $\mathcal{H}_i(\mathsf{arg}) \leftarrow (\mathsf{Enu}_i \circ \mathsf{Enc}_i \circ \mathsf{Enc}_{\mathsf{RO}})\left(\widetilde{h}_{\mathsf{ctr}}\right)$
4:     $\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))\}$
5: end if
6: return $\mathcal{H}_i(\mathsf{arg})$
```

```
$\mathcal{O}_{\mathsf{BS}}(\widetilde{\boldsymbol{w}})$:
1: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
2: $\mathsf{sr} \leftarrow (\mathsf{Enu}_3 \circ \mathsf{Enc}_{\mathsf{BS}})\left(\widetilde{h}_{\mathsf{ctr}}\right)$
3: $\boldsymbol{z} \leftarrow \mathcal{P}^{-1}(\mathsf{sk}, \widetilde{\boldsymbol{w}}; \mathsf{sr})$
4: return $\boldsymbol{z}$
```

Figure 3: Description of the random oracles $\mathcal{O}_{\mathcal{H}_i}$ and blind signature oracle $\mathcal{O}_{\mathsf{BS}}$ involved in the wrapper algorithm $\mathcal{B}_k$.

$J \neq 0$; $(J, \sigma) \longleftarrow \mathcal{B}_k(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}; \omega)\}$ and $\mathcal{W}_j = \{(\overline{\mathsf{inst}}_1, \omega, \widetilde{\boldsymbol{h}}) \in \overline{\mathsf{Prmsti}}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}} : (j, \sigma) \longleftarrow \mathcal{B}_k(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}; \omega)\}$. Then, we can write $\mathcal{W} = \cup_{j=1}^{\widetilde{\nu}} \mathcal{W}_j$. Note that $\mathcal{W}$ and $\mathcal{W}_j$'s are independent of the choice of k. So, we have the following expression for the success probability.

$$\begin{aligned} \mathsf{acc} &= \Pr_{(\overline{\mathsf{inst}}_1, \omega, \widetilde{\boldsymbol{h}}) \xleftarrow{\$} \overline{\mathsf{Prmsti}}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}}} \left[(\overline{\mathsf{inst}}_1, \omega, \widetilde{\boldsymbol{h}}) \in \mathcal{W}\right] \\ &\geq (1 - \frac{1}{|\widetilde{\mathsf{R}}|}) \cdot \varepsilon \\ &\approx \varepsilon \end{aligned} \tag{24}$$

where $\varepsilon = \mathsf{Adv}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa)$, and the term $(1 - 1/|\widetilde{\mathsf{R}}|)$ involved in the middle expression is due to the correct guess of the output of the random oracle for $\mathcal{H}_2$ in steps 6 to 8 of Algorithm 7.

Next, we design an extended forking algorithm which is given as Algorithm 8. The algorithm randomly chooses $i^* \in [\ell+1]$ as an index for a non-trivial forgery. This algorithm takes $(\overline{\mathsf{inst}}_1, \mathsf{inst}_2) \in \overline{\mathsf{Prmsti}}_1 \times \mathsf{Prmsti}_2$ as input and runs $\mathcal{B}_{i^*}$ three times on related inputs to get 3 signatures on the same message that are associated with three different sets of random values from $\widetilde{\mathsf{R}}$. This implies that when the status bit b is equal to 1, then all the signatures σ, σ' and σ'' will have the same $\boldsymbol{w}$ and $\mathbf{ct}$.

Algorithm 8 Extended Forking for PR-mSTI

```
1: procedure $\mathsf{ExtF}_{\mathcal{B}}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)$    ▷ return a status bit $b$, and three signatures $\sigma$, $\sigma'$ and $\sigma''$
2:     $i^* \xleftarrow{\$} [\ell + 1]$    ▷ guess for non-trivial index
3:     choose $\widetilde{\boldsymbol{h}} \xleftarrow{\$} \widetilde{\mathsf{R}}^{\widetilde{\nu}}$
4:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i^*}$
5:     $(J, \sigma) \longleftarrow \mathcal{B}_{i^*}(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}; \omega)$
6:     if $J = 0$ then
7:         return $(0, \epsilon, \epsilon, \epsilon)$
8:     end if
9:     pick $\widetilde{h}'_J, \ldots, \widetilde{h}'_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
10:    set $\widetilde{\boldsymbol{h}}' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J-1}, \widetilde{h}'_J, \ldots, \widetilde{h}'_{\widetilde{\nu}})$
11:    $(J', \sigma') \longleftarrow \mathcal{B}_{i^*}(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}'; \omega)$
12:    if $J \neq J'$ or $\widetilde{h}_J = \widetilde{h}'_J$ then
13:        return $(0, \epsilon, \epsilon, \epsilon)$
14:    end if
15:    pick $\widetilde{h}''_J, \ldots, \widetilde{h}''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
16:    set $\widetilde{\boldsymbol{h}}'' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J-1}, \widetilde{h}''_J, \ldots, \widetilde{h}''_{\widetilde{\nu}})$
17:    $(J'', \sigma'') \longleftarrow \mathcal{B}_{i^*}(\overline{\mathsf{inst}}_1, \widetilde{\boldsymbol{h}}''; \omega)$
18:    if $J' \neq J''$ or $\widetilde{h}_J = \widetilde{h}''_J$ or $\widetilde{h}'_J = \widetilde{h}''_J$ then
19:        return $(0, \epsilon, \epsilon, \epsilon)$
20:    end if
21:    return $(1, \sigma, \sigma', \sigma'')$
22: end procedure
```

The success probability of $\mathsf{ExtF}_{\mathcal{B}}$ can be expressed in terms of the forking probability frk as defined in

Corollary 3.10. In fact,

$$\begin{aligned}
\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) &= \mathsf{Pr}\left[b = 1 : \ (\overline{\mathsf{inst}}_1, \mathsf{inst}_2) \xleftarrow{\$} \overline{\mathsf{Prmsti}}; \ (b, \sigma, \sigma', \sigma'') \longleftarrow \mathsf{ExtF}_{\mathcal{B}}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)\right] \\
&= \sum_{j=1}^{\widetilde{\nu}} \mathsf{frk}(\mathcal{W}_j, j) \\
&= \mathsf{frk} \\
&\geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^2}{16 \cdot \widetilde{\nu}^2} - \frac{3}{|\widetilde{\mathsf{R}}|} \right) \qquad \text{[using Corollary 3.10]} \\
&\geq \varepsilon \cdot \left(\frac{\varepsilon^2}{16 \cdot \widetilde{\nu}^2} - \frac{3}{|\widetilde{\mathsf{R}}|} \right) \qquad \text{[using equation (24)} \\
&\geq \frac{\varepsilon^3}{16 \cdot \widetilde{\nu}^2} \qquad (25)
\end{aligned}$$

where we refer to Lemma 3.9 for the definition of $\mathsf{frk}(\mathcal{W}_j, j)$. Note that the random choice of i^* in step 2 of Algorithm 8 does not affect the success probability.

Hybrid Argument. First, note that the role of $\mathcal{B}_{i^*}$ is to create an environment for $\mathcal{A}$ and whenever $\mathcal{A}$ succeeds, $\mathcal{B}_{i^*}$ also produces a forgery based on the output of $\mathcal{A}$. So, if $\mathcal{B}_{i^*}$ can create a proper environment for $\mathcal{A}$, then the success will totally depend on the success of $\mathcal{A}$. Furthermore, the success of $\mathsf{ExtF}_{\mathcal{B}}$ depends on the success of $\mathcal{B}_{i^*}$ and therefore the success of $\mathcal{A}$. We will modify the extended forking algorithm so that the control of $\mathcal{O}_{\mathsf{BS}}$ gradually shifts from $\mathcal{B}_{i^*}$ to an external entity through $\mathsf{ExtF}_{\mathcal{B}}$. Thus, sk is no longer useful for $\mathsf{ExtF}_{\mathcal{B}}$ and can be omitted from its input. Also, we remove some redundancy that are not useful in handling the blind signature queries. The following hybrid argument establishes that none of these modifications changes the view of $\mathcal{A}$. Hence, the success probability of the extended forking algorithm will remain the same.

1. $\mathsf{ExtF}^{(1)}_{\mathcal{B}}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)$: This is defined in a similar manner to $\mathsf{ExtF}_{\mathcal{B}}$, except for the handling of $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$ inside the $\mathcal{O}_{\mathsf{BS}}$-oracle and the input to $\mathcal{B}_{i^*}$. More precisely, $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$ (see Remark 2.7) will now be handled by $\mathsf{ExtF}^{(1)}_{\mathcal{B}}$. In this case, $\mathcal{B}_{i^*}$ will supply $\widetilde{\boldsymbol{w}}$ (queried by $\mathcal{A}$) as well as the randomness sr involved in $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$ to $\mathsf{ExtF}^{(1)}_{\mathcal{B}}$ (see the description of $\mathcal{O}_{\mathsf{BS}}$ given below). Note that the supplied sk is now redundant and therefore omitted from the input of $\mathcal{B}_{i^*}$.

   Conceptually, there are no changes from the point of view of $\mathcal{A}$. Therefore, the success probability of the modified extended forking remains the same, i.e., $\mathsf{Succ}(\mathsf{ExtF}^{(1)}_{\mathcal{B}}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$.

```
$\underline{\mathcal{O}_{\mathsf{BS}}(\widetilde{\boldsymbol{w}})}$:
 1: $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
 2: $\mathsf{sr} \leftarrow (\mathsf{Enu}_3 \circ \mathsf{Enc}_{\mathsf{BS}}) \left(\widetilde{h}_{\mathsf{ctr}}\right)$
 3: $\boldsymbol{z} \xleftarrow{Q} \mathcal{P}^{-1}(\mathsf{sk}, \widetilde{\boldsymbol{w}}; \mathsf{sr})$ ▷ $\mathcal{B}_{i^*}$ makes query on $(\widetilde{\boldsymbol{w}}, \mathsf{sr})$ to $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$
 4: return $\boldsymbol{z}$
```

2. $\mathsf{ExtF}^{(2)}_{\mathcal{B}}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)$: This is the same as $\mathsf{ExtF}^{(1)}_{\mathcal{B}}$ except for the implementation of the $\mathcal{O}_{\mathsf{BS}}$-oracle. In this modification, $\mathcal{B}_{i^*}$ does not need to supply the randomness sr involved in $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$ to $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$. Rather, for each blind signature query $\widetilde{\boldsymbol{w}}$ submitted by $\mathcal{B}_{i^*}$, $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$ samples the fresh randomness

Algorithm 9 Extended Forking-2

```
1: procedure $\mathsf{ExtF}^{(2)}_{\mathcal{B}}(\overline{\mathsf{inst}_1}, \mathsf{inst}_2)$
2:     $\boxed{\mathsf{List}_{\mathsf{BS}} \leftarrow \emptyset \text{ and } \mathsf{ind} \leftarrow 0}$
3:     $i^* \xleftarrow{\$} [\ell + 1]$                    ▷ guess for non-trivial index
4:     choose $\widetilde{\boldsymbol{h}} \xleftarrow{\$} \widetilde{\mathsf{R}}^{\widetilde{\nu}}$
5:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i^*}$
6:     $\boxed{(J, \sigma) \longleftarrow \mathcal{B}^{\overline{\mathcal{O}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)}$
7:     if $J = 0$ then
8:         return $(0, \epsilon, \epsilon, \epsilon)$
9:     end if
10:    pick $\widetilde{h}'_J, \ldots, \widetilde{h}'_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
11:    set $\widetilde{\boldsymbol{h}}' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J-1}, \widetilde{h}'_J, \ldots, \widetilde{h}'_{\widetilde{\nu}})$
12:    $\boxed{\mathsf{ind} \leftarrow 0}$
13:    $\boxed{(J', \sigma') \longleftarrow \mathcal{B}^{\overline{\overline{\mathcal{O}}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'; \omega)}$
14:    if $J \neq J'$ or $\widetilde{h}_J = \widetilde{h}'_J$ then
15:        return $(0, \epsilon, \epsilon, \epsilon)$
16:    end if
17:    pick $\widetilde{h}''_J, \ldots, \widetilde{h}''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
18:    set $\widetilde{\boldsymbol{h}}'' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J-1}, \widetilde{h}''_J, \ldots, \widetilde{h}''_{\widetilde{\nu}})$
19:    $\boxed{\mathsf{ind} \leftarrow 0}$
20:    $\boxed{(J'', \sigma'') \longleftarrow \mathcal{B}^{\overline{\overline{\mathcal{O}}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''; \omega)}$
21:    if $J' \neq J''$ or $\widetilde{h}_J = \widetilde{h}''_J$ or $\widetilde{h}'_J = \widetilde{h}''_J$ then
22:        return $(0, \epsilon, \epsilon, \epsilon)$
23:    end if
24:    return $(1, \sigma, \sigma', \sigma'')$
25: end procedure
```

by itself. To maintain consistency of the answers of the blind signature queries that appear before the forking index, $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$ maintains a local database $\mathsf{List}_{\mathsf{BS}}$. Basically, in $\mathsf{List}_{\mathsf{BS}}$, it stores all the answers to the blind signature queries involved in the first run. While answering blind signature queries in the 2nd and 3rd runs, $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$ replies with the stored values if $\widetilde{\boldsymbol{w}}$ appears before the forking index. Otherwise, it answers by sampling fresh randomness sr. For notational simplicity, let us refer to $\mathcal{P}^{-1}(\mathsf{sk}, \cdot, \cdot)$ as $\mathcal{O}(\cdot)$. We denote the oracles used in the first run by $\overline{\mathcal{O}}(\cdot)$, and in the second and third runs by $\overline{\overline{\mathcal{O}}}(\cdot)$. Both oracles $\overline{\mathcal{O}}(\cdot)$ and $\overline{\overline{\mathcal{O}}}(\cdot)$ are handled by $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$ (see the table below). For completeness, $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$ is illustrated as Algorithm 9, where $\mathcal{B}^{\overline{\mathcal{O}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)$ is invoked in the first run and $\mathcal{B}^{\overline{\overline{\mathcal{O}}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'; \omega)$ and $\mathcal{B}^{\overline{\overline{\mathcal{O}}}}_{i^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''; \omega)$ are invoked in the second and third runs, respectively. Note that in steps 6, 13 and 20 of Algorithm 9, the superscripts $\overline{\mathcal{O}}$ and $\overline{\overline{\mathcal{O}}}$ mean that the blind signature oracle is supplied to $\mathcal{B}_{i^*}$. All changes are highlighted using bounding boxes. Again, none of these modifications affects the view of $\mathcal{A}$. Therefore, $\mathsf{Succ}(\mathsf{ExtF}^{(2)}_{\mathcal{B}}) = \mathsf{Succ}(\mathsf{ExtF}^{(1)}_{\mathcal{B}}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$.

```
$\overline{\mathcal{O}}(\widetilde{\boldsymbol{w}})$: // for 1st run
1: $\mathsf{ind} \leftarrow \mathsf{ind} + 1$
2: $\mathsf{sr} \xleftarrow{\$} (\mathbb{F}^v)^t$
3: $\boldsymbol{z} \leftarrow \mathcal{P}^{-1}(\mathsf{sk}, \widetilde{\boldsymbol{w}}; \mathsf{sr})$
4: $\mathsf{List}_{\mathsf{BS}}[\mathsf{ind}] \leftarrow \boldsymbol{z}$
5: return $\boldsymbol{z}$
```

```
$\overline{\overline{\mathcal{O}}}(\widetilde{\boldsymbol{w}})$: // for 2nd and 3rd run
1: if $\widetilde{\boldsymbol{w}}$ appears before $J$ then
2:     $\mathsf{ind} \leftarrow \mathsf{ind} + 1$
3:     return $\mathsf{List}_{\mathsf{BS}}[\mathsf{ind}]$
4: else
5:     $\mathsf{sr} \xleftarrow{\$} (\mathbb{F}^v)^t$
6:     $\boldsymbol{z} \leftarrow \mathcal{P}^{-1}(\mathsf{sk}, \widetilde{\boldsymbol{w}}; \mathsf{sr})$
7:     return $\boldsymbol{z}$
8: end if
```

3. $\mathsf{ExtF}^{(3)}_{\mathcal{B}}(\overline{\mathsf{inst}_1}, \mathsf{inst}_2)$: This is defined in a similar manner to $\mathsf{ExtF}^{(2)}_{\mathcal{B}}$, except that some redundancy is removed. Note that $\mathcal{B}_{i^*}$ received a random vector $\widetilde{\boldsymbol{h}}$ of size $\widetilde{\nu} = \nu + \ell$, where the ℓ components of $\widetilde{\boldsymbol{h}}$ are used to synthesise randomness in handling blind signature queries by $\mathcal{A}$. Since $\mathcal{B}_{i^*}$ is now supplied with a blind signature oracle, these additional ℓ components of $\widetilde{\boldsymbol{h}}$ are no longer required. Note that $\mathsf{Enc}_{\mathsf{RO}}$ and $\mathsf{Enc}_{\mathsf{BS}}$ were only used to split $\widetilde{\boldsymbol{h}}$ by sending ν and ℓ many components to the index sets R and R_3, respectively. Therefore, the encoding functions Enu_3 and $\mathsf{Enc}_{\mathsf{BS}}$ are also no longer required. In addition, $\mathcal{B}_{i^*}$ can simply be given ν many random elements from R. So, essentially, we remove $\widetilde{\mathsf{R}}$,

$\mathsf{Enc}_{\mathsf{RO}}$, and $\mathsf{Enc}_{\mathsf{BS}}$. That is, the input of $\mathcal{B}_{i^*}$ will now become $(\mathsf{inst}_1, \boldsymbol{h})$, where $\boldsymbol{h} \in \mathsf{R}^{\nu}$ and the random oracles inside $\mathcal{B}_{i^*}$ are now implemented as described in the table below. Likewise, some of the steps in $\mathsf{ExtF}_{\mathcal{B}}^{(2)}$ will be modified. For completeness, $\mathsf{ExtF}_{\mathcal{B}}^{(3)}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)$ is illustrated as Algorithm 10. All changes are highlighted using bounding boxes.

Algorithm 10 Extended Forking-3

```
1: procedure $\mathsf{ExtF}_{\mathcal{B}}^{(3)}(\overline{\mathsf{inst}}_1, \mathsf{inst}_2)$
2:     $\mathsf{List}_{\mathsf{BS}} \leftarrow \emptyset$ and $\mathsf{ind} \leftarrow 0$
3:     $i^* \xleftarrow{\$} [\ell + 1]$                    ▷ guess for non-trivial index
4:     choose $\boxed{\boldsymbol{h} \xleftarrow{\$} \mathsf{R}^{\nu}}$
5:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i^*}$
6:     $\boxed{(J, \sigma) \longleftarrow \mathcal{B}_{i^*}^{\overline{\mathcal{O}}}(\mathsf{inst}_1, \boldsymbol{h}; \omega)}$
7:     if $J = 0$ then
8:         return $(0, \epsilon, \epsilon, \epsilon)$
9:     end if
10:    pick $\boxed{h'_J, \ldots, h'_{\nu} \xleftarrow{\$} \mathsf{R}}$
11:    set $\boxed{h' = (h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_{\nu})}$
12:    $\mathsf{ind} \leftarrow 0$
13:    $\boxed{(J', \sigma') \longleftarrow \mathcal{B}_{i^*}^{\overline{\overline{\mathcal{O}}}}(\mathsf{inst}_1, \boldsymbol{h}'; \omega)}$
14:    if $\boxed{J \neq J' \text{ or } h_J = h'_J}$ then
15:        return $(0, \epsilon, \epsilon, \epsilon)$
16:    end if
17:    pick $\boxed{h''_J, \ldots, h''_{\nu} \xleftarrow{\$} \mathsf{R}}$
18:    set $\boxed{h'' = (h_1, \ldots, h_{J-1}, h''_J, \ldots, h''_{\nu})}$
19:    $\mathsf{ind} \leftarrow 0$
20:    $\boxed{(J'', \sigma'') \longleftarrow \mathcal{B}_{i^*}^{\overline{\overline{\mathcal{O}}}}(\mathsf{inst}_1, \boldsymbol{h}''; \omega)}$
21:    if $J' \neq J''$ or $\boxed{h_J = h''_J}$ or $\boxed{h'_J = h''_J}$ then
22:        return $(0, \epsilon, \epsilon, \epsilon)$
23:    end if
24:    return $(1, \sigma, \sigma', \sigma'')$
25: end procedure
```

```
$\mathcal{O}_{\mathcal{H}_i}(\mathsf{arg})$: // for $i = 1, 2$
1: if $\mathcal{H}_i(\mathsf{arg})$ is not defined then
2:     $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
3:     $\boxed{\mathcal{H}_i(\mathsf{arg}) \leftarrow (\mathsf{Enu}_i \circ \mathsf{Enc}_i)(h_{\mathsf{ctr}})}$
4:     $\mathsf{List} \leftarrow \mathsf{List} \cup \{(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))\}$
5: end if
6: return $\mathcal{H}_i(\mathsf{arg})$
```

Again, none of these modifications affects the view of $\mathcal{A}$. Therefore, the success probability of $\mathsf{ExtF}_{\mathcal{B}}^{(3)}$ remains the same, that is, $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(3)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(2)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$.

4. $\mathsf{ExtF}_{\mathcal{B}}^{(4)}(\mathsf{inst}_1, \mathsf{inst}_2)$: This is the same as $\mathsf{ExtF}_{\mathcal{B}}^{(3)}$ except for the implementation of $\mathcal{P}^{-1}(sk, \cdot, \cdot)$ inside $\overline{\mathcal{O}}(\cdot)$ and $\overline{\overline{\mathcal{O}}}(\cdot)$. In this modification, $\mathsf{ExtF}_{\mathcal{B}}^{(4)}$ will get access to an external oracle $\mathcal{O}(\cdot)$ (actually the helper oracle involved in the PR-mSTI problem). This means that to implement $\overline{\mathcal{O}}(,)$ and $\overline{\overline{\mathcal{O}}}(\cdot)$, $\mathsf{ExtF}_{\mathcal{B}}^{(4)}$ simply forwards the query $\widetilde{\boldsymbol{w}}$ from $\mathcal{B}_{i^*}$ to the external oracle $\mathcal{O}()$ and gets the answer $\boldsymbol{z}$. Note that $\mathsf{List}_{\mathsf{BS}}$ still needs to be maintained for consistency. Also, note that sk is omitted from the input of $\mathsf{ExtF}_{\mathcal{B}}^{(4)}$ as it is no longer useful. Again, this modification does not change the view of $\mathcal{A}$. Therefore, $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(4)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(3)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$.

5. $\mathsf{ExtF}_{\mathcal{B}}^{(5)}(\mathsf{inst}_1, \mathsf{inst}_2)$: This is defined similarly to $\mathsf{ExtF}_{\mathcal{B}}^{(4)}$ except for the following. First, note that, so far, the second input inst_2 was not used. In this modification, the hash index, where M_{i^*} appears as an argument of the $\mathcal{H}_1$-query, is guessed randomly. Let j^* be the guess. Then, the j^*-th value of $\boldsymbol{h}$ is programmed through the encoding functions as well as the input inst_2 (see Steps 4 to 6 of Algorithm 11, highlighted using bounding boxes). We emphasise that $\hbar^*$ is uniformly distributed

over R due to property (5). Again, this modification does not change the view of $\mathcal{A}$. Therefore, $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(5)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{(4)}) = \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})$. Henceforth, we refer to $\mathsf{ExtF}_{\mathcal{B}}^{(5)}$ as $\mathsf{ExtF}_{\mathcal{B}}^{\mathcal{O}}$ to mean that $\mathsf{ExtF}_{\mathcal{B}}$ has access to the external oracle $\mathcal{O}(\cdot)$ (i.e., the oracle involved in the PR-mSTI problem).

Algorithm 11 Extended Forking-5

```
1: procedure $\mathsf{ExtF}_{\mathcal{B}}^{(5)}(\mathsf{inst}_1, \mathsf{inst}_2)$
2:     $\mathsf{List_{BS}} \leftarrow \emptyset$ and $\mathsf{ind} \leftarrow 0$
3:     $i^* \xleftarrow{\$} [\ell+1]$                              ▷ guess for non-trivial index
4:     $\boxed{j^* \xleftarrow{\$} [\nu]}$                              ▷ guess for hash index of $\mathrm{M}_{i^*}$
5:     pick $\boxed{\hbar^* \xleftarrow{\$} (\mathsf{Enu}_1 \circ \mathsf{Enc}_1)^{-1}(\mathsf{inst}_2)}$
6:     choose $\boldsymbol{h} \xleftarrow{\$} \mathsf{R}^{\nu}$ such that $\boxed{h_{j^*} = \hbar^*}$
7:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i^*}$
8:     $(J, \sigma) \longleftarrow \mathcal{B}_{i^*}^{\mathcal{O}}(\mathsf{inst}_1, \boldsymbol{h}; \omega)$
9:     if $J = 0$ then
10:        return $(0, \epsilon, \epsilon, \epsilon)$
11:    end if
12:    pick $h'_J, \ldots, h'_\nu \xleftarrow{\$} \mathsf{R}$
13:    set $h' = (h_1, \ldots, h_{J-1}, h'_J, \ldots, h'_\nu)$
14:    $\mathsf{ind} \leftarrow 0$
15:    $(J', \sigma') \longleftarrow \mathcal{B}_{i^*}^{\overline{\overline{\mathcal{O}}}}(\mathsf{inst}_1, \boldsymbol{h}'; \omega)$
16:    if $J \neq J'$ or $h_J = h'_J$ then
17:        return $(0, \epsilon, \epsilon, \epsilon)$
18:    end if
19:    pick $h''_J, \ldots, h''_\nu \xleftarrow{\$} \mathsf{R}$
20:    set $h'' = (h_1, \ldots, h_{J-1}, h''_J, \ldots, h''_\nu)$
21:    $\mathsf{ind} \leftarrow 0$
22:    $(J'', \sigma'') \longleftarrow \mathcal{B}_{i^*}^{\overline{\overline{\mathcal{O}}}}(\mathsf{inst}_1, \boldsymbol{h}''; \omega)$
23:    if $J' \neq J''$ or $h_J = h''_J$ or $h'_J = h''_J$ then
24:        return $(0, \epsilon, \epsilon, \epsilon)$
25:    end if
26:    return $(1, \sigma, \sigma', \sigma'')$
27: end procedure
```

PR-mSTI Solver. Finally, we design a PR-mSTI solver which will take advantage of the extended forking algorithm $\mathsf{ExtF}_{\mathcal{B}}^{\mathcal{O}}$ (Algorithm 11) and solves a given problem instance. Let $((\mathcal{P}, \mathcal{R}), \boldsymbol{w}^*) \in \mathsf{Prmsti}_1 \times \mathsf{Prmsti}_2$ be a given random instance of the PR-mSTI problem. Then, the PR-mSTI solver algorithm Simu is described as Algorithm 12.

Algorithm 12 PR-mSTI Solver

```
1: procedure $\mathsf{Simu}^{\mathcal{O}}(\mathcal{P}, \mathcal{R}, \boldsymbol{w}^*)$          ▷ Output $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ such that $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \boldsymbol{w}^*$
2:     set $\mathsf{inst}_1 = (\mathcal{P}, \mathcal{R})$ and $\mathsf{inst}_2 = \boldsymbol{w}^*$
3:     $(b, \sigma, \sigma', \sigma'') \longleftarrow \mathsf{ExtF}_{\mathcal{B}}^{\mathcal{O}}(\mathsf{inst}_1, \mathsf{inst}_2)$
4:     if $b = 0$ then
5:         abort
6:     end if
7:     parse $\sigma, \sigma'$ and $\sigma''$ as $\sigma = (\mathbf{ct}, \mathbf{ch}, \mathbf{rs})$, $\sigma' = (\mathbf{ct}, \mathbf{ch}', \mathbf{rs}')$ and $\sigma'' = (\mathbf{ct}, \mathbf{ch}'', \mathbf{rs}'')$
8:     find an $i \in [r]$ such that $\mathsf{ch}_i \neq \mathsf{ch}'_i \neq \mathsf{ch}''_i \neq \mathsf{ch}_i$
9:     if such $i$ is not found then
10:        abort
11:    end if
12:    set $\pi = (\mathsf{ct}_i, \mathsf{ch}_i, \mathsf{rs}_i)$, $\pi' = (\mathsf{ct}_i, \mathsf{ch}'_i, \mathsf{rs}'_i)$ and $\pi'' = (\mathsf{ct}_i, \mathsf{ch}''_i, \mathsf{rs}''_i)$
13:    $(\boldsymbol{z}, \bar{\boldsymbol{z}}) \longleftarrow \mathsf{Extr}(\pi, \pi', \pi'')$          ▷ Such an extractor exists due to Lemma B.2
14:    if $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) \neq \boldsymbol{w}^*$ then
15:        abort
16:    else
17:        return $(\boldsymbol{z}, \bar{\boldsymbol{z}})$
18:    end if
19: end procedure
```

Using Proposition A.2, we get the probability of reaching step 13 of Algorithm 12 without abort is $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}^{\mathcal{O}}) \cdot \mathsf{f}$, where $\mathsf{f} = 1 - (7/9)^r$ is close to 1. Note that $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ (computed in step 13 of Algorithm 12) will always satisfy $\mathcal{P}(\boldsymbol{z}) + \mathcal{R}(\bar{\boldsymbol{z}}) = \mathcal{H}_1(\mathrm{M}_{i^*})$ as the underlying commitment scheme is computationally binding. To ensure $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ to be a valid witness of the given problem instance, we have to show the following.

1. $\widetilde{\boldsymbol{w}}^* = \boldsymbol{w}^* - \mathcal{R}(\bar{\boldsymbol{z}})$ was never queried to the external oracle $\mathcal{O}$.

2. $\mathcal{H}_1(\mathrm{M}_{i^*}) = \mathsf{inst}_2 = \boldsymbol{w}^*$.

Note that $(\boldsymbol{z},\bar{\boldsymbol{z}})$ is the witness involved in the i^*-th signature returned by $\mathcal{A}$. So, if i^* is the index of a non-trivial forgery, then $\widetilde{\boldsymbol{w}}^*$ will never appear as an argument to $\mathcal{O}$ (see Definition 5.2). Therefore, the first requirement can be guaranteed with probability at least $1/(\ell+1)$. Furthermore, with probability at least $1/\nu$, the message M_{i^*} (associated with non-trivial forgery) appears at the index j^* and $\mathcal{H}_1(\mathrm{M}_{i^*})$ gets the programmed value $(\mathsf{Enu}_1 \circ \mathsf{Enc}_1)(\hbar^*)$. Hence, by step 5 of Algorithm 11, we can write $\mathcal{H}_1(\mathrm{M}_{i^*}) = (\mathsf{Enu}_1 \circ \mathsf{Enc}_1)(\hbar^*) = \mathsf{inst}_2 = \boldsymbol{w}^*$. Therefore, the success probability of finding a valid solution is

$$\begin{aligned}
\mathsf{Adv}^{\text{PR-mSTI}}_{\mathsf{Simu}}(\kappa) &\geq \frac{1}{\ell+1}\cdot\frac{1}{\nu}\cdot\mathsf{Succ}(\mathsf{ExtF}^{\mathcal{O}}_{\mathcal{B}})\cdot\mathsf{f} - \mathsf{Adv}^{\text{Binding}}_{\mathsf{Simu}}(\kappa)\\
&= \frac{1}{\ell+1}\cdot\frac{1}{\nu}\cdot\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}})\cdot\mathsf{f} - \mathsf{Adv}^{\text{Binding}}_{\mathsf{Simu}}(\kappa)\\
&\geq \frac{1}{\ell+1}\cdot\frac{1}{\nu}\cdot\frac{\varepsilon^3}{16\cdot\widetilde{\nu}^2}\cdot\mathsf{f} - \mathsf{Adv}^{\text{Binding}}_{\mathsf{Simu}}(\kappa) \qquad (26)
\end{aligned}$$

where the last step follows from equation (25). ☐

Algorithm 13 Wrapper for CMQ

```
1: procedure $\mathcal{B}_{k_1,k_2}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}})$
2:     $(\mathsf{pk}', \mathsf{sk}') \longleftarrow \mathsf{UOV.KeyGen}(\kappa)$
3:     run $\mathsf{ck} \xleftarrow{\$} \mathsf{CSetup}(\kappa)$
4:     pick $r \in \mathbb{N}$ such that $(2/3)^r = \mathsf{negl}(\kappa)$
5:     $\mathsf{List} \leftarrow \emptyset$ and $\mathsf{ctr} \leftarrow 0$    ▷ List stores tuples of the form $(\mathsf{ctr}, \mathsf{arg}, \mathcal{H}_i(\mathsf{arg}))$ for $i \in [2]$
6:     $\left\{(\mathrm{M}_i, (\mathbf{ct}^{(i)}, \mathbf{rs}^{(i)}))_{i\in[\ell+1]}\right\} \longleftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{all}}}((\mathsf{pk}', \mathsf{inst}_1), \mathsf{ck}, r)$    ▷ oracles' description are the same as defined in Figure 3
7:     if $J_{k_1}$ or $J_{k_2}$ is not defined then
8:         return $(0, 0, \epsilon, \epsilon)$
9:     end if
10:    for $i = 1$ to $\ell + 1$ do
11:        $\boldsymbol{w}_i \xleftarrow{Q} \mathcal{H}_1(\mathrm{M}_i)$
12:        $\mathbf{ch}^{(i)} \xleftarrow{Q} \mathcal{H}_2(\boldsymbol{w}_i, \mathbf{ct}^{(i)})$
13:        set $\sigma_i = (\mathbf{ct}^{(i)}, \mathbf{ch}^{(i)}, \mathbf{rs}^{(i)})$
14:    end for
15:    if $\mathsf{Ver}(\mathsf{pk}, \mathrm{M}_i, \sigma_i) = 0$ for some $i \in [\ell]$ or $\mathrm{M}_i = \mathrm{M}_j$ for some $i, j \in [\ell+1]$ with $i \neq j$ then    ▷ where $\mathsf{pk} = ((\mathsf{pk}', \mathsf{inst}_1), \mathsf{ck})$
16:        return $(0, 0, \epsilon, \epsilon)$
17:    end if
18:    set $J_1 = \min\{J_{k_1}, J_{k_2}\}$ and $J_2 = \max\{J_{k_1}, J_{k_2}\}$
19:    set $\sigma_1 = \sigma_{J_1}$ and $\sigma_2 = \sigma_{J_2}$
20:    return $(J_1, J_2, \sigma_1, \sigma_2)$
21: end procedure
```

Next, we show that if there is a collision set (in the sense of Definition 5.1), then one can construct a solver for the CMQ problem. Let Cmq denote the set of all instances of the CMQ problem. Write $\mathsf{Cmq} = \mathsf{Cmq}_1 \times \mathsf{Cmq}_2$, where $\mathsf{Cmq}_1 = \mathcal{P}(\mathbb{F}^{\bar{n}}, \mathbb{F}^m)$ and $\mathsf{Cmq}_2 = \mathbb{F}^m$. Let $(\mathcal{R}, \boldsymbol{y}^*) \xleftarrow{\$} \mathsf{Cmq}_1 \times \mathsf{Cmq}_2$ be an instance of the CMQ problem. Let $\mathrm{M}_{i_1^*}$ and $\mathrm{M}_{i_2^*}$ be two messages for which a collision occurs, i.e., $\widetilde{\boldsymbol{w}}_{i_1^*} = \widetilde{\boldsymbol{w}}_{i_2^*}$. Our wrapper algorithm $\mathcal{B}_{i_1^*,i_2^*}$ (see Algorithm 13) will output two indices $J_{i_1^*}$, $J_{i_2^*}$ of $\mathcal{H}_2$-oracle and the associated signatures $\sigma_{i_1^*}$ and $\sigma_{i_2^*}$. Note that the wrapper algorithm itself samples the UOV key-pair. Observe that the wrapper algorithm ensures $J_{i_1^*} < J_{i_2^*}$. Our extended forking algorithm rewinds the wrapper algorithm twice at $J_{i_1^*}$ and twice at $J_{i_2^*}$. That is, there are five runs in total of the wrapper algorithm $\mathcal{B}_{i_1^*,i_2^*}$. When forking at the index $J_{i_1^*}$ (resp. $J_{i_2^*}$) is considered, then the signatures associated with the index $J_{i_2^*}$ (resp. $J_{i_1^*}$) are ignored. So, the initial run of $\mathcal{B}_{i_1^*,i_2^*}$ followed by two rewinding at $J_{i_1^*}$ (resp. $J_{i_2^*}$) will give three signatures for the same message $\mathrm{M}_{i_1^*}$ (resp. $\mathrm{M}_{i_2^*}$). These three signatures essentially commit the same argument $(\boldsymbol{w}_{i_1^*}, \mathbf{ct}^{(i_1^*)})$ (resp. $(\boldsymbol{w}_{i_2^*}, \mathbf{ct}^{(i_2^*)})$) via the hash function $\mathcal{H}_2$, where $\boldsymbol{w}_{i_1^*} = \mathcal{H}_1(\mathrm{M}_{i_1^*})$ (resp. $\boldsymbol{w}_{i_2^*} = \mathcal{H}_1(\mathrm{M}_{i_2^*})$). To calculate the success probability of the multi-index forking, we use Corollary 3.12.

Next, find three component transcripts π_1, π_1' and π_1'' with different challenges (similarly to the PR-mSTI solver) from the first tuple of signatures (ie, associated with $\mathrm{M}_{i_1^*}$) and then apply the extractor to obtain $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1)$ such that $\mathcal{P}(\boldsymbol{z}_1) + \mathcal{R}(\bar{\boldsymbol{z}}_1) = \boldsymbol{w}_{i_1^*}$. Similarly, from other tuples of signatures (i.e., associated with $\mathrm{M}_{i_2^*}$), we obtain $(\boldsymbol{z}_2, \bar{\boldsymbol{z}}_2)$ such that $\mathcal{P}(\boldsymbol{z}_2)+\mathcal{R}(\bar{\boldsymbol{z}}_2) = \boldsymbol{w}_{i_2^*}$. By collision, we have $\mathcal{P}(\boldsymbol{z}_1) = \widetilde{\boldsymbol{w}}_{i_1^*} = \widetilde{\boldsymbol{w}}_{i_2^*} = \mathcal{P}(\boldsymbol{z}_2)$.

Thus, we have $\mathcal{R}(\bar{\boldsymbol{z}}_1) - \mathcal{R}(\bar{\boldsymbol{z}}_2) = \boldsymbol{w}_{i_1^*} - \boldsymbol{w}_{i_2^*}$. By appropriate programming of $\mathcal{H}_1$ via encoding functions and the target $\boldsymbol{y}^*$, we guarantee $\boldsymbol{w}_{i_1^*} - \boldsymbol{w}_{i_2^*} = \boldsymbol{y}^*$. More formally, the reduction is given as follows.

Lemma 5.2. *Suppose an OMF-attacker produces $(\ell + 1)$ many message-signature pairs after making ℓ many blind signature queries in the random oracle model. If the CMQ problem (cf. Definition 2.16) is intractable and the underlying commitment scheme is computationally binding (cf. Definition 2.3), then the forgeries form a collision-free set (cf. Definition 5.1).*

Proof. Let $\mathcal{A}$ be an OMF-attacker that produces a set $\mathfrak{S}$ of $(\ell + 1)$ message-signature pairs after making ℓ many blind signature queries. Suppose $\mathfrak{S}$ is not collision-free. That is, there are at least two distinct indices $i_1^*, i_2^* \in [\ell + 1]$ such that $\widetilde{\boldsymbol{w}}_{i_1^*} = \widetilde{\boldsymbol{w}}_{i_2^*}$ (cf. Definition 5.1). Then, we solve the CMQ problem using $\mathcal{A}$ as a subroutine. We use the same encoding functions and their properties as were used in Theorem 5.1.

Wrapper Algorithm. For $k_1, k_2 \in [\ell + 1]$ with $k_1 \neq k_2$, we construct a wrapper algorithm $\mathcal{B}_{k_1,k_2}$ as Algorithm 13 which takes $(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}) \in \mathsf{Cmq}_1 \times \widetilde{\mathsf{R}}^{\widetilde{\nu}}$ as input and creates an environment for $\mathcal{A}$, and when $\mathcal{A}$ returns $(\ell + 1)$ message-signature pairs, $\mathcal{B}_{k_1,k_2}$ returns a tuple $(J_1, J_2, \sigma_1, \sigma_2)$. Similarly to the proof of Theorem 5.1, here σ_1, σ_2 are the side outputs and $J_1, J_2 \in [0, \widetilde{\nu}]$ are the forking indices. The wrapper $\mathcal{B}_{k_1,k_2}$ handles the random oracle queries and the blind signature queries in a manner similar to that done in Figure 3. Note that the wrapper algorithm knows the secret key as it samples the UOV key-pair. It is expected that the adversary $\mathcal{A}$ makes queries to $\mathcal{H}_2$-oracle corresponding to the messages M_{k_1} and M_{k_2}. If the related query to $\mathcal{H}_2$-oracle, say, for M_{k_1} was not made, then we say that J_{k_1} (forking-index) is not defined. If J_{k_1} or J_{k_2} is not defined, then Algorithm 13 returns $(0, 0, \epsilon, \epsilon)$ (see steps 7 to 9). By the definition of $\mathcal{B}_{k_1,k_2}$, the indices J_1 and J_2 involved in the output can be both zero or non-zero. Furthermore, when they are non-zero, then $J_1 < J_2$ due to step 18 of Algorithm 13.

Let $\mathcal{W} = \{(\mathsf{inst}_1, \omega, \widetilde{\boldsymbol{h}}) \in \mathsf{Cmq}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}} : J_1 < J_2;\ (J_1, J_2, \sigma_1, \sigma_2) \longleftarrow \mathcal{B}_{k_1,k_2}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)\}$ and $\mathcal{W}_{j_1,j_2} = \{(\mathsf{inst}_1, \omega, \widetilde{\boldsymbol{h}}) \in \mathsf{Cmq}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}} : (j_1, j_2, \sigma_1, \sigma_2) \longleftarrow \mathcal{B}_{k_1,k_2}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)\}$ for $j_1, j_2 \in [\widetilde{\nu}]$. Then $\mathcal{W} = \bigcup_{\substack{j_1,j_2 \in [\widetilde{\nu}] \\ j_1 < j_2}} \mathcal{W}_{j_1,j_2}$. Note that the definitions of $\mathcal{W}$ and $\mathcal{W}_{j_1,j_2}$ are independent of the choice of $k_1, k_2 \in [\ell+1]$. Here, the success of $\mathcal{B}_{k_1,k_2}$ captures the fact that "$J_1 < J_2$". If acc denotes the success probability of $\mathcal{B}_{k_1,k_2}$, then using Corollary 3.12, we have the following.

$$\begin{aligned} \mathsf{acc} &= \Pr_{(\mathsf{inst}_1, \omega, \widetilde{\boldsymbol{h}}) \xleftarrow{\$} \mathsf{Cmq}_1 \times \Omega \times \widetilde{\mathsf{R}}^{\widetilde{\nu}}} \left[(\mathsf{inst}_1, \omega, \widetilde{\boldsymbol{h}}) \in \mathcal{W} \right] \\ &\geq \left(1 - \frac{2}{|\widetilde{\mathsf{R}}|}\right) \cdot \varepsilon \\ &\approx \varepsilon \end{aligned} \tag{27}$$

where $\varepsilon = \mathsf{Adv}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa)$ and the term $(1 - 2/|\widetilde{\mathsf{R}}|)$ involved in the middle expression is due to the correct guess of the output of the random oracle for $\mathcal{H}_2$ in steps 7 to 9 of Algorithm 13.

Extended Forking Algorithm. The extended forking algorithm (Algorithm 14) samples a pair of indices $i_1^*, i_2^* \in [\ell + 1]$ as a guess of a collision. This algorithm takes $(\mathsf{inst}_1, \mathsf{inst}_2) \in \mathsf{Cmq}_1 \times \mathsf{Cmq}_2$ as input and runs $\mathcal{B}_{i_1^*,i_2^*}$ five times on related inputs to get two tuples of signatures $(\sigma_1, \sigma_1', \sigma_1'')$ and $(\sigma_2, \sigma_2''', \sigma_2'''')$. In the first run, $\mathcal{B}_{i_1^*,i_2^*}$ outputs $(J_1, J_2, \sigma_1, \sigma_2)$. Then $\mathcal{B}_{i_1^*,i_2^*}$ is rewound at the index J_1 twice (i.e., the second and third runs) to obtain $(J_1', J_2', \sigma_1', \sigma_2')$ and $(J_1'', J_2'', \sigma_1'', \sigma_2'')$. Similarly, $\mathcal{B}_{i_1^*,i_2^*}$ is rewound at the index J_2 twice (that is, fourth and fifth runs) to get $(J_1''', J_2''', \sigma_1''', \sigma_2''')$ and $(J_1'''', J_2'''', \sigma_1'''', \sigma_2'''')$. If all the required conditions are met, then $(\sigma_1, \sigma_1', \sigma_1'')$ and $(\sigma_2, \sigma_2''', \sigma_2'''')$ are three valid signatures for messages $\mathrm{M}_{i_1^*}$ and $\mathrm{M}_{i_2^*}$, respectively.

Similarly to Theorem 5.1, the success probability of $\mathsf{ExtF}_{\mathcal{B}}$ can be expressed in terms of the forking

Algorithm 14 Extended Forking for CMQ

```
1: procedure $\mathsf{ExtF}_{\mathcal{B}}(\mathsf{inst}_1, \mathsf{inst}_2)$
2:     $i_1^*, i_2^* \xleftarrow{\$} [\ell+1]$                ▷ guess for collision indices
3:     choose $\widetilde{\boldsymbol{h}} \xleftarrow{\$} \widetilde{\mathsf{R}}^{\widetilde{\nu}}$
4:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i_1^*, i_2^*}$
5:     $(J_1, J_2, \sigma_1, \sigma_2) \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)$                ▷ 1st run
6:     if $J_1 = 0$ or $J_2 = 0$ then
7:         return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
8:     end if
9:     pick $\widetilde{h}'_{J_1}, \ldots, \widetilde{h}'_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
10:    set $\widetilde{\boldsymbol{h}}' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_1-1}, \widetilde{h}'_{J_1}, \ldots, \widetilde{h}'_{\widetilde{\nu}})$
11:    $(J_1', J_2', \sigma_1', \sigma_2') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'; \omega)$                ▷ 2nd run
12:    if $J_1 \neq J_1'$ or $\boxed{J_2 \neq J_2'}$ or $\widetilde{h}_{J_1} = \widetilde{h}'_{J_1}$ then
13:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
14:    end if
15:    pick $\widetilde{h}''_{J_1}, \ldots, \widetilde{h}''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
16:    set $\widetilde{\boldsymbol{h}}'' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_1-1}, \widetilde{h}''_{J_1}, \ldots, \widetilde{h}''_{\widetilde{\nu}})$
17:    $(J_1'', J_2'', \sigma_1'', \sigma_2'') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''; \omega)$                ▷ 3rd run
18:    if $J_1' \neq J_1''$ or $\boxed{J_2' \neq J_2''}$ or $\widetilde{h}_{J_1} = \widetilde{h}''_{J_1}$ or $\widetilde{h}'_{J_1} = \widetilde{h}''_{J_1}$ then
19:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
20:    end if
21:    pick $\widetilde{h}'''_{J_2}, \ldots, \widetilde{h}'''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
22:    set $\widetilde{\boldsymbol{h}}''' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_2-1}, \widetilde{h}'''_{J_2}, \ldots, \widetilde{h}'''_{\widetilde{\nu}})$
23:    $(J_1''', J_2''', \sigma_1''', \sigma_2''') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'''; \omega)$                ▷ 4th run
24:    if $\boxed{J_1 \neq J_1'''}$ or $J_2 \neq J_2'''$ or $\widetilde{h}_{J_2} = \widetilde{h}'''_{J_2}$ then
25:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
26:    end if
27:    pick $\widetilde{h}''''_{J_2}, \ldots, \widetilde{h}''''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
28:    set $\widetilde{\boldsymbol{h}}'''' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_2-1}, \widetilde{h}''''_{J_2}, \ldots, \widetilde{h}''''_{\widetilde{\nu}})$
29:    $(J_1'''', J_2'''', \sigma_1'''', \sigma_2'''') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''''; \omega)$                ▷ 5th run
30:    if $\boxed{J_1''' \neq J_1''''}$ or $J_2''' \neq J_2''''$ or $\widetilde{h}_{J_2} = \widetilde{h}''''_{J_2}$ or $\widetilde{h}'''_{J_2} = \widetilde{h}''''_{J_2}$
    then
31:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
32:    end if
33:    return $(1, (\sigma_1, \sigma_1', \sigma_1''), (\sigma_2, \sigma_2''', \sigma_2''''))$
34: end procedure
```

probability frk as follows.

$$\begin{aligned}
\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) &= \mathsf{Pr}\left[b = 1 : \ (\mathsf{inst}_1, \mathsf{inst}_2) \xleftarrow{\$} \mathsf{Cmq}; \ (b, (\sigma_1, \sigma_1', \sigma_1''), (\sigma_2, \sigma_2''', \sigma_2'''')) \longleftarrow \mathsf{ExtF}_{\mathcal{B}}(\mathsf{inst}_1, \mathsf{inst}_2)\right] \\
&= \sum_{\substack{j_1, j_2 \in [\nu] \\ j_1 < j_2}} \mathsf{frk}(\mathcal{W}_{j_1, j_2}, j_1, j_2) \\
&= \mathsf{frk} \\
&\geq \mathsf{acc} \cdot \left(\frac{\mathsf{acc}^4}{16 \cdot (\widetilde{\nu}^2 - \widetilde{\nu})^4} - \frac{6}{|\widetilde{\mathsf{R}}|} \right) \qquad \text{[using Corollary 3.12]} \\
&\geq \varepsilon \cdot \left(\frac{\varepsilon^4}{16 \cdot (\widetilde{\nu}^2 - \widetilde{\nu})^4} - \frac{6}{|\widetilde{\mathsf{R}}|} \right) \qquad \text{[using equation (27)]} \\
&\approx \frac{\varepsilon^5}{16 \cdot (\widetilde{\nu}^2 - \widetilde{\nu})^4} \qquad (28)
\end{aligned}$$

where we refer to Lemma 3.11 for the definition of $\mathsf{frk}(\mathcal{W}_{j_1, j_2}, j_1, j_2)$.

Note that the checks that are performed inside the boxes of Algorithm 14 are sort of redundant. In fact, the success probability is expected to improve if they are removed. This, however, requires a more involved forking abstraction and associated analysis which we leave as an interesting open question.

Modified Extended Forking Algorithm $\overline{\mathsf{ExtF}_{\mathcal{B}}}(\mathsf{inst}_1, \mathsf{inst}_2)$: Now, we modify the extended forking algorithm to embed part of the problem instance inst_2 into a specific index of $\widetilde{\boldsymbol{h}}$. Basically, first, we guess two indices $j_1^*, j_2^* \in [\widetilde{\nu}]$, where $\mathrm{M}_{i_1^*}$ and $\mathrm{M}_{i_2^*}$ could appear in the $\mathcal{H}_1$-oracle. Then, we programme j_1^*-th entry of $\widetilde{\boldsymbol{h}}$ using our encoding functions, the j_2^*-th entry of $\widetilde{\boldsymbol{h}}$ and inst_2. These changes are highlighted using bounding boxes in Algorithm 15. We emphasise that these modifications do not change the view of $\mathcal{A}$ as $\widetilde{h}_{j_2^*}$ remains uniform over $\widetilde{\mathsf{R}}$. In fact, by the properties of the encoding functions, $\boldsymbol{w}_2^*$ is uniformly distributed over $\mathbb{F}^m$, so $\mathsf{inst}_2 + \boldsymbol{w}_2^*$ is uniformly distributed over $\mathbb{F}^m$.[9] Therefore, by the properties of the encoding functions, the programmed value $\widetilde{h}_{j_2^*}$ is uniformly distributed over $\widetilde{\mathsf{R}}$. Hence, $\mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) = \mathsf{Succ}(\overline{\mathsf{ExtF}_{\mathcal{B}}})$.

Algorithm 15 Modified Extended Forking for CMQ

```
1: procedure $\overline{\mathsf{ExtF}_{\mathcal{B}}}(\mathsf{inst}_1, \mathsf{inst}_2)$
2:     $i_1^*, i_2^* \xleftarrow{\$} [\ell+1]$                      ▷ guess for collision indices
3:     $\boxed{j_1^*, j_2^* \xleftarrow{\$} [\widetilde{\nu}]}$                ▷ guess for hash indices of $\mathrm{M}_{i_1^*}$, $\mathrm{M}_{i_1^*}$
4:     $\boxed{\hbar_2^* \xleftarrow{\$} \widetilde{\mathsf{R}}}$
5:     $\boxed{\text{compute } \boldsymbol{w}_2^* = (\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}})\,(\hbar_2^*)}$
6:     $\boxed{\text{pick } \hbar_1^* \xleftarrow{\$} (\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}})^{-1}\,(\mathsf{inst}_2 + \boldsymbol{w}_2^*)}$
7:     $\boxed{\text{choose } \widetilde{\boldsymbol{h}} \xleftarrow{\$} \widetilde{\mathsf{R}}^{\widetilde{\nu}} \text{ such that } \widetilde{h}_{j_1^*} = \hbar_1^* \text{ and } \widetilde{h}_{j_2^*} = \hbar_2^*}$
8:     pick random coin $\omega \xleftarrow{\$} \Omega$ for $\mathcal{B}_{i_1^*, i_2^*}$
9:     $(J_1, J_2, \sigma_1, \sigma_2) \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}; \omega)$          ▷ 1st run
10:    if $J_1 = 0$ or $J_2 = 0$ then
11:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
12:    end if
13:    pick $\widetilde{h}'_{J_1}, \ldots, \widetilde{h}'_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
14:    set $\widetilde{\boldsymbol{h}}' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_1-1}, \widetilde{h}'_{J_1}, \ldots, \widetilde{h}'_{\widetilde{\nu}})$
15:    $(J_1', J_2', \sigma_1', \sigma_2') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'; \omega)$          ▷ 2nd run
16:    if $J_1 \neq J_1'$ or $J_2 \neq J_2'$ or $\widetilde{h}_{J_1} = \widetilde{h}'_{J_1}$ then
17:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
18:    end if
19:    pick $\widetilde{h}''_{J_1}, \ldots, \widetilde{h}''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
20:    set $\widetilde{\boldsymbol{h}}'' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_1-1}, \widetilde{h}''_{J_1}, \ldots, \widetilde{h}''_{\widetilde{\nu}})$
21:    $(J_1'', J_2'', \sigma_1'', \sigma_2'') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''; \omega)$          ▷ 3rd run
22:    if $J_1' \neq J_1''$ or $J_2' \neq J_2''$ or $\widetilde{h}_{J_1} = \widetilde{h}''_{J_1}$ or $\widetilde{h}'_{J_1} = \widetilde{h}''_{J_1}$ then
23:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
24:    end if
25:    pick $\widetilde{h}'''_{J_2}, \ldots, \widetilde{h}'''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
26:    set $\widetilde{\boldsymbol{h}}''' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_2-1}, \widetilde{h}'''_{J_2}, \ldots, \widetilde{h}'''_{\widetilde{\nu}})$
27:    $(J_1''', J_2''', \sigma_1''', \sigma_2''') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}'''; \omega)$          ▷ 4th run
28:    if $J_1 \neq J_1'''$ or $J_2 \neq J_2'''$ or $\widetilde{h}_{J_2} = \widetilde{h}'''_{J_2}$ then
29:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
30:    end if
31:    pick $\widetilde{h}''''_{J_2}, \ldots, \widetilde{h}''''_{\widetilde{\nu}} \xleftarrow{\$} \widetilde{\mathsf{R}}$
32:    set $\widetilde{\boldsymbol{h}}'''' = (\widetilde{h}_1, \ldots, \widetilde{h}_{J_2-1}, \widetilde{h}''''_{J_2}, \ldots, \widetilde{h}''''_{\widetilde{\nu}})$
33:    $(J_1'''', J_2'''', \sigma_1'''', \sigma_2'''') \longleftarrow \mathcal{B}_{i_1^*, i_2^*}(\mathsf{inst}_1, \widetilde{\boldsymbol{h}}''''; \omega)$          ▷ 5th run
34:    if $J_1''' \neq J_1''''$ or $J_2''' \neq J_2''''$ or $\widetilde{h}_{J_2} = \widetilde{h}''''_{J_2}$ or $\widetilde{h}'''_{J_2} = \widetilde{h}''''_{J_2}$ then
35:        return $(0, (\epsilon, \epsilon, \epsilon), (\epsilon, \epsilon, \epsilon))$
36:    end if
37:    return $(1, (\sigma_1, \sigma_1', \sigma_1''), (\sigma_2, \sigma_2''', \sigma_2''''))$
38: end procedure
```

CMQ Solver. Our CMQ problem solver is illustrated in Algorithm 16. Here, the solver Simu takes a CMQ problem instance $(\mathcal{R}, \boldsymbol{y}^*)$ as input, runs $\overline{\mathsf{ExtF}_{\mathcal{B}}}$ and returns a solution of the problem. Similarly to the PR-mSTI solver, the probability of reaching step 15 of Algorithm 16 without abort (using Proposition A.2) is $\mathsf{Succ}(\overline{\mathsf{ExtF}_{\mathcal{B}}}) \cdot \mathsf{f}^2$, where $\mathsf{f} = 1 - (7/9)^r$ is close to 1. Note that $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1)$ and $(\boldsymbol{z}_2, \bar{\boldsymbol{z}}_2)$ (computed in step 15 of Algorithm 16) always satisfy the following equations, as the underlying commitment scheme is computationally binding.

$$\mathcal{P}(\boldsymbol{z}_1) + \mathcal{R}(\bar{\boldsymbol{z}}_1) = \mathcal{H}_1(\mathrm{M}_{i_1^*}) \tag{29}$$
$$\mathcal{P}(\boldsymbol{z}_2) + \mathcal{R}(\bar{\boldsymbol{z}}_2) = \mathcal{H}_1(\mathrm{M}_{i_2^*}). \tag{30}$$

1. If i_1^* and i_2^* are correctly guessed, then $\mathcal{P}(\boldsymbol{z}_1) = \mathcal{P}(\boldsymbol{z}_2)$ (using the hypothesis that $\mathfrak{S}$ is a collision set. In fact, a collision will be found for the indices i_1^* and i_2^*. So, $\mathcal{P}(\boldsymbol{z}_1) = \boldsymbol{w}_{i_1^*} - \mathcal{R}(\bar{\boldsymbol{z}}_1) = \widetilde{\boldsymbol{w}}_{i_1^*} = \widetilde{\boldsymbol{w}}_{i_2^*} = \boldsymbol{w}_{i_2^*} - \mathcal{R}(\bar{\boldsymbol{z}}_2) = \mathcal{P}(\boldsymbol{z}_2)$, where we write $\boldsymbol{z}_1, \boldsymbol{z}_2, \bar{\boldsymbol{z}}_1$ and $\bar{\boldsymbol{z}}_2$ for $\boldsymbol{z}_{i_1^*}, \boldsymbol{z}_{i_2^*}, \bar{\boldsymbol{z}}_{i_1^*}$ and $\bar{\boldsymbol{z}}_{i_2^*}$, respectively. Then by subtracting the above two equations (29) and (30), we have $\mathcal{R}(\boldsymbol{z}_1) - \mathcal{R}(\boldsymbol{z}_2) = \mathcal{H}_1(\mathrm{M}_{i_1^*}) - \mathcal{H}_1(\mathrm{M}_{i_2^*})$.

2. If j_1^* and j_2^* are the correct guesses, then we have the following situation. By the construction of $\overline{\mathsf{ExtF}_{\mathcal{B}}}$, $\mathcal{H}_1(\mathrm{M}_{i_2^*}) = (\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}})\,(\widetilde{h}_{j_2^*}) = \boldsymbol{w}_2^*$ and $\mathcal{H}_1(\mathrm{M}_{i_1^*}) = (\mathsf{Enu}_1 \circ \mathsf{Enc}_1 \circ \mathsf{Enc}_{\mathsf{RO}})\,(\widetilde{h}_{j_1^*}) = \mathsf{inst}_2 + \boldsymbol{w}_2^*$ (due to step 6 of Algorithm 15). Then, we have $\mathcal{R}(\boldsymbol{z}_1) - \mathcal{R}(\boldsymbol{z}_2) = \mathcal{H}_1(\mathrm{M}_{i_1^*}) - \mathcal{H}_1(\mathrm{M}_{i_2^*}) = \boldsymbol{y}^*$.

Therefore, if $\overline{\mathsf{ExtF}_{\mathcal{B}}}$ can make the guesses correctly, then $(\bar{\boldsymbol{z}}_1, \bar{\boldsymbol{z}}_2)$ will be a correct solution. Hence, the

[9]Note that the uniformity of $\mathsf{inst}_2 + \boldsymbol{w}_2^*$ can also be argued using the uniformity of inst_2.

Algorithm 16 CMQ Solver

```
1: procedure Simu($\mathcal{R}, \boldsymbol{y}^*$)    ▷ output $(\bar{\boldsymbol{z}}_1, \bar{\boldsymbol{z}}_2)$ such that $\mathcal{R}(\bar{\boldsymbol{z}}_1) - \mathcal{R}(\bar{\boldsymbol{z}}_2) = \boldsymbol{y}^*$
2:     set $\mathsf{inst}_1 = \mathcal{R}$ and $\mathsf{inst}_2 = \boldsymbol{y}^*$
3:     $(b, (\sigma_1, \sigma_1', \sigma_1''), (\sigma_2, \sigma_2''', \sigma_2''''))\longleftarrow\overline{\mathsf{ExtF}_{\mathcal{B}}}(\mathsf{inst}_1, \mathsf{inst}_2)$
4:     if $b = 0$ then
5:         abort
6:     end if
7:     parse $\sigma_1, \sigma_1'$ and $\sigma_2''$ as $\sigma_1 = (\mathbf{ct}^{(1)}, \mathbf{ch}^{(1)}, \mathbf{rs}^{(1)})$, $\sigma_1' = (\mathbf{ct}^{(1)}, \mathbf{ch}^{(2)}, \mathbf{rs}^{(2)})$ and $\sigma_1'' = (\mathbf{ct}^{(1)}, \mathbf{ch}^{(3)}, \mathbf{rs}^{(3)})$
8:     parse $\sigma_2, \sigma_2'''$ and $\sigma_2''''$ as $\sigma_2 = (\mathbf{ct}^{(4)}, \mathbf{ch}^{(4)}, \mathbf{rs}^{(4)})$, $\sigma_2''' = (\mathbf{ct}^{(4)}, \mathbf{ch}^{(5)}, \mathbf{rs}^{(5)})$ and $\sigma_2'''' = (\mathbf{ct}^{(4)}, \mathbf{ch}^{(6)}, \mathbf{rs}^{(6)})$
9:     find $i_1, i_2 \in [r]$ such that $\mathsf{ch}_{i_1}^{(1)} \neq \mathsf{ch}_{i_1}^{(2)} \neq \mathsf{ch}_{i_1}^{(3)} \neq \mathsf{ch}_{i_1}^{(1)}$ and $\mathsf{ch}_{i_2}^{(4)} \neq \mathsf{ch}_{i_2}^{(5)} \neq \mathsf{ch}_{i_2}^{(6)} \neq \mathsf{ch}_{i_2}^{(4)}$
10:    if such $i_1$ and $i_2$ are not found then
11:        abort
12:    end if
13:    set $\pi_1 = (\mathsf{ct}_{i_1}^{(1)}, \mathsf{ch}_{i_1}^{(1)}, \mathsf{rs}_{i_1}^{(1)})$, $\pi_1' = (\mathsf{ct}_{i_1}^{(1)}, \mathsf{ch}_{i_1}^{(2)}, \mathsf{rs}_{i_1}^{(2)})$ and $\pi_1'' = (\mathsf{ct}_{i_1}^{(1)}, \mathsf{ch}_{i_1}^{(3)}, \mathsf{rs}_{i_1}^{(3)})$
14:    set $\pi_2 = (\mathsf{ct}_{i_2}^{(4)}, \mathsf{ch}_{i_2}^{(4)}, \mathsf{rs}_{i_2}^{(4)})$, $\pi_2''' = (\mathsf{ct}_{i_2}^{(4)}, \mathsf{ch}_{i_2}^{(5)}, \mathsf{rs}_{i_2}^{(5)})$ and $\pi_2'''' = (\mathsf{ct}_{i_2}^{(4)}, \mathsf{ch}_{i_2}^{(6)}, \mathsf{rs}_{i_2}^{(6)})$
15:    $(\boldsymbol{z}_1, \bar{\boldsymbol{z}}_1)\longleftarrow\mathsf{Extr}(\pi_1, \pi_1', \pi_1'')$ and $(\boldsymbol{z}_2, \bar{\boldsymbol{z}}_2)\longleftarrow\mathsf{Extr}(\pi_2, \pi_2''', \pi_2'''')$    ▷ Such an extractor exists due to Lemma B.5
16:    if $\mathcal{R}(\bar{\boldsymbol{z}}_1) - \mathcal{R}(\bar{\boldsymbol{z}}_2) \neq \boldsymbol{y}^*$ then
17:        abort
18:    else
19:        return $(\bar{\boldsymbol{z}}_1, \bar{\boldsymbol{z}}_2)$
20:    end if
21: end procedure
```

success probability of finding a valid solution is

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{Simu}}^{\mathrm{CMQ}}(\kappa) &\geq \frac{1}{(\ell+1)^2} \cdot \frac{1}{(\widetilde{\nu})^2} \cdot \mathsf{Succ}(\overline{\mathsf{ExtF}_{\mathcal{B}}}) \cdot \mathsf{f}^2 - \mathsf{Adv}_{\mathsf{Simu}}^{\mathrm{Binding}}(\kappa) \\
&= \frac{1}{(\ell+1)^2} \cdot \frac{1}{(\widetilde{\nu})^2} \cdot \mathsf{Succ}(\mathsf{ExtF}_{\mathcal{B}}) \cdot \mathsf{f}^2 - \mathsf{Adv}_{\mathsf{Simu}}^{\mathrm{Binding}}(\kappa) \\
&\geq \frac{1}{(\ell+1)^2} \cdot \frac{1}{(\widetilde{\nu})^2} \cdot \frac{\varepsilon^5}{16 \cdot (\widetilde{\nu}^2 - \widetilde{\nu})^4} \cdot \mathsf{f}^2 - \mathsf{Adv}_{\mathsf{Simu}}^{\mathrm{Binding}}(\kappa) \qquad (31)
\end{aligned}$$

where the last step follows equation (27). □

Corollary 5.3. *Suppose that there exists an adversary $\mathcal{A}$ who can break the OMF-security of the MBSS described in Section 5.1 in the random oracle model. Then, using $\mathcal{A}$ as a subroutine, we can create PPT algorithms $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ for breaking the CMQ problem, the PR-mSTI problem and the computational binding property of the underlying commitment scheme respectively with the following relation of the respective advantages:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{OMF}}(\kappa) \leq C_1 \cdot \left(\mathsf{Adv}_{\mathcal{B}_1}^{\mathrm{CMQ}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_3}^{\mathrm{Binding}}(\kappa)\right)^{1/5} + C_2 \cdot \left(\mathsf{Adv}_{\mathcal{B}_2}^{\mathrm{PR\text{-}mSTI}}(\kappa) + \mathsf{Adv}_{\mathcal{B}_3}^{\mathrm{Binding}}(\kappa)\right)^{1/3}$$

where $C_1 = \left(16 \cdot (\ell+1)^2 \cdot \widetilde{\nu}^2 \cdot (\widetilde{\nu}^2 - \widetilde{\nu})^4 / f^2\right)^{1/5}$, $C_2 = \left(16 \cdot (\ell+1) \cdot \nu \cdot \widetilde{\nu}^2 / f\right)^{1/3}$, $\widetilde{\nu} = \nu + \ell$, and ℓ and ν are respectively the numbers of blind signature queries and hash queries, and $f = 1 - (7/9)^r$.

Proof. Follows from the security bounds in equations 26 and 31. □

6 Conclusion

The paper has proposed modular security reductions for 3 and 5-pass MQDSS [CHR+16] and a slightly modified version of the multivariate blind signature scheme from [PSM17]. To the best of our knowledge, this is the first work to provide concrete security analysis for digital signature and blind signature schemes in the MQ-setting. We have introduced several new forking abstractions that serve as a core tool in security reductions. The proposed forking algorithms are likely to find further applications, in particular in the MQ setting. Although the (modified) MBSS uses UOV as a black-box primitive signature scheme, other UOV variants, like MAYO, QR-UOV, and SNOVA, can also be used. Therefore, finding a suitable UOV-based signature and judicious parameter settings for the MBSS, based on our concrete security analysis, could be an interesting question of practical relevance.

For blind signature security, we have introduced two new hardness assumptions in the non-linear MQ-setting that do not seem to have any analogue in the more familiar linear settings like RSA. Studying the classical/quantum intractability of these new assumptions as well as finding their further cryptographic applications are some of the interesting potential future works.

The security reductions discussed in this paper rely on the rewinding technique in the random oracle model. In general, security proofs that involve rewinding or random oracles do not naturally extend to the quantum setting due to limitations such as the no-cloning theorem and reliance on the forking lemma. Therefore, studying concrete security reductions of such schemes in the quantum random oracle model, especially in the context of Fiat-Shamir-style conversion, remains an interesting open problem.

References

[AF22] Thomas Attema and Serge Fehr. Parallel Repetition of $(k_1, \ldots, k_\mu)$-Special-Sound Multi-Round Interactive Proofs. In *CRYPT*, volume 4117 of *LNCS*, pages 415–443. Springer, 2022.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In *15th ACM conference on Computer and communications security*, pages 449–458. SIAM, 2008.

[Beu21] Ward Beullens. MAYO: Practical Post-quantum Signatures from Oil-and-Vinegar Maps. In *SAC*, volume 13203 of *LNCS*, pages 355–376. Springer, 2021.

[Beu22] Ward Beullens. Breaking Rainbow Takes a Weekend on a Laptop. In *CRYPTO*, volume 13508 of *LNCS*, pages 464–479. Springer, 2022.

[Beu24] Ward Beullens. Multivariate Blind Signatures Revisited. Cryptology ePrint Archive, Report 2024/720, 2024. `http://eprint.iacr.org/`.

[BFR23] Ryad Benadjila and Thibauld Feneuil and Matthieu Rivain MQ on my Mind: Post-Quantum Signatures from the Non-Structured Multivariate Quadratic Problem. Cryptology ePrint Archive, Report 2023/1719, 2023. `http://eprint.iacr.org/`.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *13th ACM conference on Computer and communications security*, pages 390–399. Association for Computing Machinery, New York, United States, 2006.

[BNP$^+$03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology*, 16(3):255–275, 2003.

[BPW12] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. *Journal of Cryptology*, 25(1):57–115, 2012.

[BTZ22] Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger Security for Non-Interactive Threshold Signatures: BLS and FROST. Cryptology ePrint Archive, Report 2022/833, 2022. `http://eprint.iacr.org/`.

[CDP21] Sanjit Chatterjee, Akansha Dimri, and Tapas Pandit. Identity-Based Signature and Extended Forking Algorithm in the Multivariate Quadratic Setting. In *INDOCRYPT*, volume 13143 of *LNCS*, pages 387–412. Springer, 2021.

[Cha82] David Chaum. Blind Signatures for Untraceable Payments. In *CRYPTO*, pages 199–203. Plenum Press, New York, USA, 1982, 1982.

[CHL05] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 302–321. Springer, 2005.

[CHR$^+$16] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. From 5-Pass MQ-Based Identification to MQ-Based Signatures. In *ASIACRYPT*, volume 10032 of *LNCS*, pages 135–165. Springer, 2016.

[CK16] Sanjit Chatterjee and Chethan Kamath. A Closer Look at Multiple Forking: Leveraging (In)Dependence for a Tighter Bound. *Algorithmica*, 74(4):1321–1362, 2016.

[CKK12] Sanjit Chatterjee, Chethan Kamath, and Vikas Kumar. Galindo-Garcia Identity-Based Signature Revisited. In *ICISC*, volume 7839 of *LNCS*, pages 456–471. Springer, 2012.

[CKM$^+$16] Sanjit Chatterjee, Neal Koblitz, Alfred Menezes, and Palash Sarkar. Another look at tightness II: practical issues in cryptography. In *Mycrypt*, volume 10311 of *LNCS*, pages 21–55. Springer, 2016.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, volume 2045 of *LNCS*, pages 93–118. Springer, 2001.

[CMS11] Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another Look at Tightness. In *Selected Areas in Cryptography*, volume 7118 of *LNCS*, pages 293–319. Springer, 2011.

[CMW12] Sherman S. M. Chow, Changshe Ma, and Jian Weng. Zero-Knowledge Argument for Simultaneous Discrete Logarithms. *Algorithmica*, 64(2):246–266, 2012.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In *CRYPTO*, volume 1880 of *LNCS*, pages 229–235. Springer, 2000.

[DS05] Jintai Ding and Dieter Schmidt. Rainbow, a New Multivariable Polynomial Signature Scheme. In *Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 164–175. Springer, 2005.

[FIH$^+$23] Hiroki Furue and Yasuhiko Ikematsu and Fumitaka Hoshino and Tsuyoshi Takagi and Kan Yasuda and Toshiyuki Miyazawa and Tsunekazu Saito and Akira Nagai. QR-UOV. Technical report, National Institute of Standards and Technology, 2023. Available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures `https://csrc.nist.gov/projects/pqc-dig-sig/round-1-additional-signatures`.

[Fis06] Marc Fischlin. Round-Optimal Composable Blind Signatures in the Common Reference String Model. In *CRYPT*, volume 4117 of *LNCS*, pages 60–77. Springer, 2006.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[GG09] David Galindo and Flavio D. Garcia. A Schnorr-Like Lightweight Identity-Based Signature Scheme. In Bart Preneel, editor, *AFRICACRYPT*, volume 5580 of *LNCS*, pages 135–148. Springer, 2009.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A Modular Treatment of Blind Signatures from Identification Schemes. In *EUROCRYPT*, volume 11478 of *LNCS*, pages 345–375. Springer, 2019.

[Jen06] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193, 1906.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 206–222. Springer, 1999.

[KZ20] Daniel Kales and Greg Zaverucha. An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes. In *Cryptology and Network Security*, volume 12579 of *LNCS*, pages 3–22. Springer, 2020.

[Maj21] Aalo Majumdar. Security of post-quantum multivariate blind signature scheme: Revisited and improved. Master's thesis, Indian Institute Science Bangalore, 2021.

[NIS19] National Institute of Standards and Technology. Post-quantum crypto project (Second Round Submission). `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions`, 2019. Accessed: 2022-08-16.

[NIS20] National Institute of Standards and Technology. Post-quantum crypto project (Third Round Submission). `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions`, 2020. Accessed: 2022-08-16.

[NIS23] National Institute of Standards and Technology. Post-Quantum Cryptography: Digital Signature Schemes. `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`, 2023. Accessed: 2024-06-9.

[NIS25] National Institute of Standards and Technology. Post-Quantum Cryptography: Digital Signature Schemes. `https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures`, 2025. Accessed: 2025-05-16.

[OAR$^+$07] Daniel Ortiz-Arroyo and Francisco Rodriguez-Henriquez. Yet Another Improvement over the Mu-Varadharajan e-voting Protocol. *Computer Standards & Interfaces*, 29(4):471–480, 2007.

[Oka92] Tatsuaki Okamoto. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *CRYPTO*, volume 740 of *LNCS*, pages 31–53. Springer, 1992.

[Pat97] Jacques Patarin. The Oil and Vinegar Algorithm for Signatures. In *Dagstuhl Workshop on Cryptography*, 1997.

[PS96a] David Pointcheval and Jacques Stern. Provably Secure Blind Signature Schemes. In *ASIACRYPT*, volume 1163 of *LNCS*, pages 252–265. Springer, 1996.

[PS96b] David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In *EUROCRYPT*, volume 1070 of *LNCS*, pages 387–398. Springer, 1996.

[PS97] David Pointcheval and Jacques Stern:. New Blind Signatures Equivalent to Factorization (extended abstract). In *4th ACM conference on Computer and communications security*, pages 92–99. ACM Press, 1997.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PSM17] Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A Practical Multivariate Blind Signature Scheme. In *Financial Cryptography and Data Security*, volume 10322 of *LNCS*, pages 437–454. Springer, 2017.

[Sch89] C. P. Schnorr. Efficient identification and signatures for smartcards. In *CRYPTO*, volume 435 of *LNCS*, pages 239–252. Springer, 1989.

[SSH11] Koichi Sakumoto, Taizo Shirai, and Harunaga Hiwatari. Public-Key Identification Schemes Based on Multivariate Quadratic Polynomials. In *CRYPTO*, volume 6841 of *LNCS*, pages 706–723. Springer, 2011.

[WCD$^+$24] Lih-Chung Wang and Chun-Yen Chou and Jintai Ding and Yen-Liang Kuan and Jan Adriaan Leegwater and Ming-Siou Li and Bo-Shu Tseng and Po-En Tseng and Chia-Chun Wang A Note on the SNOVA Security. Cryptology ePrint Archive, Report 2024/1517, 2024. `http://eprint.iacr.org/`.

[Zav12] G.M. Zaverucha. Hybrid encryption in the multi-user setting. Cryptology ePrint Archive, Report 2012/159, 2012. `http://eprint.iacr.org/`.

A Some Essential Results

The following proposition follows from the Jensen's inequality.

Proposition A.1. *Let a and n be positive integers. Let $x_1, \ldots, x_n \geq 0$ be real numbers. Then*

$$\sum_{i=1}^{n} x_i^a \geq \frac{1}{n^{a-1}} \left(\sum_{i=1}^{n} x_i \right)^a .$$

Proof. When $a = 1$, then the inequality becomes equality. Now, we assume that $a \geq 2$. Let $\mathcal{X}$ be a random variable such that $\mathsf{Pr}[\mathcal{X} = x_i] = 1/n$ for all $i \in [n]$. Then, we have

$$E[\mathcal{X}^a] = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^a \text{ and } (E[\mathcal{X}])^a = \frac{1}{n^a} \left(\sum_{i=1}^{n} x_i \right)^a .$$

Consider the following function $f(x) = x^a$ for all real $x \geq 0$. One can easily check that f is a convex function for all non-negative real values of a. Then, by Jensen's inequality [Jen06], we have

$$E[f(\mathcal{X})] \geq f(E[\mathcal{X}]) \implies E[\mathcal{X}^a] \geq (E[\mathcal{X}])^a \implies \sum_{i=1}^{n} x_i^a \geq \frac{1}{n^{a-1}} \left(\sum_{i=1}^{n} x_i \right)^a .$$

□

The following propositions are required to argue the security of MQDSS and MBSS.

Proposition A.2. *Let r be a positive integer. If $\mathbf{ch}, \mathbf{ch}'$ and $\mathbf{ch}''$ are chosen uniformly and independently from $\{0, 1, 2\}^r$, then we can find with probability at least $1-(7/9)^r$ an $i \in [r]$ such that $\mathsf{ch}_i \neq \mathsf{ch}'_i \neq \mathsf{ch}''_i \neq \mathsf{ch}_i$.*

Proof. For each $i \in [r]$, let Event_i denote $\mathsf{ch}_i \neq \mathsf{ch}'_i \neq \mathsf{ch}''_i \neq \mathsf{ch}_i$. Using a simple counting argument, we can write $\mathsf{Pr}[\mathsf{Event}_i] = 6/27 = 2/9$ for any i. Now, the probability of its complement is given by

$$\begin{aligned} \mathsf{Pr}[\neg\mathsf{Event}_i] &= \mathsf{Pr}\left[\mathsf{ch}_i = \mathsf{ch}'_i \vee \mathsf{ch}_i = \mathsf{ch}''_i \vee \mathsf{ch}'_i = \mathsf{ch}''_i\right] \\ &= 1 - \mathsf{Pr}[\mathsf{Event}_i] = 7/9. \end{aligned}$$

Since the components of $\mathbf{ch}, \mathbf{ch}'$ and $\mathbf{ch}''$ are chosen uniformly at random from $\{0, 1, 2\}$, we can write

$$\mathsf{Pr}[\exists i \in [r] \text{ s.t } \mathsf{Event}_i = \mathsf{true}] = 1 - \mathsf{Pr}[\neg\mathsf{Event}_1 \wedge \cdots \wedge \neg\mathsf{Event}_r] = 1 - (7/9)^r.$$

□

Proposition A.3. *Let r be a positive integer and $\mathbb{F}$ be a set of size q. Let $\mathbf{ch}_1, \mathbf{ch}'_1 \xleftarrow{\$} \mathbb{F}^r$ and $\mathbf{ch}_2, \mathbf{ch}'_2 \xleftarrow{\$} \{0, 1\}^r$. Then, we can find with probability at least $1 - (\frac{1}{2} - \frac{1}{2q})^r$ an $i \in [r]$ such that $\mathsf{ch}_{1,i} \neq \mathsf{ch}'_{1,i}$ and $\mathsf{ch}_{2,i} \neq \mathsf{ch}'_{2,i}$.*

Proof. For each $i \in [r]$, let Event_i denote the event "$\mathsf{ch}_{1,i} \neq \mathsf{ch}'_{1,i}$ and $\mathsf{ch}_{2,i} \neq \mathsf{ch}'_{2,i}$". Note that each component $\mathsf{ch}_{1,i}, \mathsf{ch}'_{1,i}, \mathsf{ch}_{2,i}$ and $\mathsf{ch}'_{2,i}$ is chosen independently and uniformly from the respective spaces. So, $\mathsf{Pr}\left[\mathsf{ch}_{1,i} \neq \mathsf{ch}'_{1,i} \wedge \mathsf{ch}_{2,i} \neq \mathsf{ch}'_{2,i}\right] = \mathsf{Pr}\left[\mathsf{ch}_{1,i} \neq \mathsf{ch}'_{1,i}\right] \cdot \mathsf{Pr}\left[\mathsf{ch}_{2,i} \neq \mathsf{ch}'_{2,i}\right] = (1 - \frac{1}{q}) \cdot (1 - \frac{1}{2}) = (\frac{1}{2} - \frac{1}{2q})$. Therefore, $\mathsf{Pr}[\exists i \in [r] : \mathsf{Event}_i = \mathsf{true}] = 1 - \mathsf{Pr}[\mathsf{Event}_1 \wedge \cdots \wedge \mathsf{Event}_r] = 1 - (\frac{1}{2} - \frac{1}{2q})^r$. □

B IDS in MQ-Setting

In this section, we reproduce the 3-pass IDS and the 5-pass IDS of [SSH11] and also discuss their properties.

3-Pass IDS. Here we first recall the 3-pass identification scheme of Sakumoto et al. [SSH11]. Choose $(\mathcal{P}, \boldsymbol{s}) \xleftarrow{\$} \mathcal{P}(\mathbb{F}^n, \mathbb{F}^m) \times \mathbb{F}^n$ and set $\boldsymbol{v} = \mathcal{P}(\boldsymbol{s})$. Run $\mathsf{ck} \longleftarrow \mathsf{CSetup}(\kappa)$, where $\mathcal{C} = (\mathsf{CSetup}, \mathsf{Commit}, \mathsf{Open})$ is an efficient commitment scheme. The public key and the secret key are given by $\mathsf{pk} = (\mathcal{P}, \boldsymbol{v}, \mathsf{ck})$ and $\mathsf{sk} = \boldsymbol{s}$, respectively. For simplicity of notation, we often omit ck from pk. Let $\mathsf{G} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}^m$ be the polar form of $\mathcal{P}$. The interaction between a prover P and a verifier V is shown in Figure 4.

$\underline{\mathsf{P}((\mathcal{P}, \boldsymbol{v}), \boldsymbol{s})\text{:}}$ $\qquad\qquad$ $\underline{\mathsf{V}(\mathcal{P}, \boldsymbol{v})\text{:}}$

pick $\boldsymbol{a}_0, \boldsymbol{b}_0 \xleftarrow{\$} \mathbb{F}^n$ and $\boldsymbol{c}_0 \xleftarrow{\$} \mathbb{F}^m$
set $\boldsymbol{a}_1 = \boldsymbol{s} - \boldsymbol{a}_0$ and $\boldsymbol{b}_1 = \boldsymbol{a}_0 - \boldsymbol{b}_0$
$\boldsymbol{c}_1 = \mathcal{P}(\boldsymbol{a}_0) - \boldsymbol{c}_0$
$\mathsf{ct}_0 \longleftarrow \mathsf{Commit}(\boldsymbol{a}_1 || \mathsf{G}(\boldsymbol{b}_0, \boldsymbol{a}_1) + \boldsymbol{c}_0)$
$\mathsf{ct}_1 \longleftarrow \mathsf{Commit}(\boldsymbol{b}_0 || \boldsymbol{c}_0)$
$\mathsf{ct}_2 \longleftarrow \mathsf{Commit}(\boldsymbol{b}_1 || \boldsymbol{c}_1)$
set $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{ct}_2)$

$\xrightarrow{\mathsf{ct}}$

$\mathsf{ch} \xleftarrow{\$} \mathsf{ChS}$

$\xleftarrow{\mathsf{ch}}$

if $\mathsf{ch} = 0$, set $\mathsf{rs} = (\boldsymbol{a}_0, \boldsymbol{b}_1, \boldsymbol{c}_1)$
if $\mathsf{ch} = 1$, set $\mathsf{rs} = (\boldsymbol{a}_1, \boldsymbol{b}_1, \boldsymbol{c}_1)$
if $\mathsf{ch} = 2$, set $\mathsf{rs} = (\boldsymbol{a}_1, \boldsymbol{b}_0, \boldsymbol{c}_0)$

$\xrightarrow{\mathsf{rs}}$

if $\mathsf{ch} = 0$, parse rs as $(\boldsymbol{a}_0, \boldsymbol{b}_1, \boldsymbol{c}_1)$ and check,
if $\mathsf{ct}_1 \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_0 - \boldsymbol{b}_1 || \mathcal{P}(\boldsymbol{a}_0) - \boldsymbol{c}_1)$
and $\mathsf{ct}_2 \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_1 || \boldsymbol{c}_1)$

if $\mathsf{ch} = 1$, parse rs as $(\boldsymbol{a}_1, \boldsymbol{b}_1, \boldsymbol{c}_1)$ and check,
if $\mathsf{ct}_0 \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_1 || \boldsymbol{v} - \mathcal{P}(\boldsymbol{a}_1) - \mathsf{G}(\boldsymbol{b}_1, \boldsymbol{a}_1) - \boldsymbol{c}_1 + \mathcal{P}(\boldsymbol{0}))$
and $\mathsf{ct}_2 \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_1 || \boldsymbol{c}_1)$

if $\mathsf{ch} = 2$, parse rs as $(\boldsymbol{a}_1, \boldsymbol{b}_0, \boldsymbol{c}_0)$ and check,
if $\mathsf{ct}_0 \overset{?}{=} \mathsf{Commit}(\boldsymbol{a}_1 || \mathsf{G}(\boldsymbol{b}_0, \boldsymbol{a}_1) + \boldsymbol{c}_0)$
and $\mathsf{ct}_1 \overset{?}{=} \mathsf{Commit}(\boldsymbol{b}_0 || \boldsymbol{c}_0)$

Figure 4: Illustration of 3-pass IDS of Sakumoto et al. [SSH11].

Sakumoto et al. [SSH11] showed that their 3-pass IDS is statistical zero-knowledge against any cheating verifier which essentially implies the following result.

Lemma B.1 (Statistical HVZK)**.** *The 3-pass IDS described in Figure 4 is statistical HVZK if the underlying commitment scheme has statistical hiding property.*

Lemma B.2 (Extractor)**.** *Assume that the underlying commitment scheme is computationally binding. Then, the 3-pass IDS described in Figure 4 satisfies the 3-special soundness (cf. Definition 2.13).*

Proof. The proof essentially follows from that of Theorem 3 of [SSH11]. □

Lemma B.3 (Knowledge error)**.** *Suppose the MQ-problem is intractable and the underlying commitment scheme is computationally binding. Then, the 3-pass IDS presented in Figure 4 is sound (cf. Definition 2.10) with knowledge error 2/3.*

Proof. First, we show the existence of a cheating prover P who can impersonate (without knowing the witness $\boldsymbol{s}$) with probability 2/3. In fact, the cheating prover P will simply follow the protocol given in Figure 4 with some random $\boldsymbol{s}'$ as a witness. Note that P always successfully impersonates when $\mathsf{ch} \neq 1$ as $\boldsymbol{v} = \mathcal{P}(\boldsymbol{s})$ is not involved in the verification. When $\mathsf{ch} = 1$, the prover P fails to convince the verifier as $\boldsymbol{v}$ is now involved in the verification process. Hence, the success probability of P is 2/3.

Now, suppose there exists an adversary $\mathcal{A}$ who can impersonate with probability $2/3 + \epsilon$ for some non-negligible ϵ. That means that whatever the challenge $\mathsf{ch} \in \{0, 1, 2\}$, the adversary $\mathcal{A}$ can cheat with probability at least ϵ. If we rewind $\mathcal{A}$ on the same commitment ct, but with three different challenges $\mathsf{ch} = 0$, $\mathsf{ch} = 1$ and $\mathsf{ch} = 2$, we will get three accepting transcripts $\pi_0 = (\mathsf{ct}, 0, \mathsf{rs}_0)$, $\pi_1 = (\mathsf{ct}, 1, \mathsf{rs}_1)$ and $\pi_2 = (\mathsf{ct}, 2, \mathsf{rs}_2)$ with probability at least ϵ. Now, apply the extractor Extr defined in Lemma B.2 on input (π_0, π_1, π_2) and get the witness $\boldsymbol{s}$ which is a contradiction to the intractability of the MQ-problem[10]. This completes the proof. □

5-Pass IDS. We now illustrate the 5-pass IDS of Sakumoto et al. [SSH11]. The description of the key-pair is the same as that of the 3-pass IDS. The interaction between a prover P and V is shown in Figure 5. Note that here $\mathsf{ch}_1 = \alpha, \mathsf{rs}_1 = (\boldsymbol{a}_1, \boldsymbol{c}_1), \mathsf{ch}_2 = \mathsf{ch}$ and $\mathsf{rs}_2 = \mathsf{rs}$.

$\underline{\mathsf{P}((\mathcal{P}, \boldsymbol{v}), \boldsymbol{s})}$:		$\underline{\mathsf{V}(\mathcal{P}, \boldsymbol{v})}$:
pick $\boldsymbol{a}_0, \boldsymbol{b}_0 \xleftarrow{\$} \mathbb{F}^n$ and $\boldsymbol{c}_0 \xleftarrow{\$} \mathbb{F}^m$ set $\boldsymbol{a}_1 = \boldsymbol{s} - \boldsymbol{a}_0$ $\mathsf{ct}_0 \longleftarrow \mathsf{Commit}(\boldsymbol{a}_0 \| \boldsymbol{b}_0 \| \boldsymbol{c}_0)$ $\mathsf{ct}_1 \longleftarrow \mathsf{Commit}(\boldsymbol{a}_1 \| \mathsf{G}(\boldsymbol{b}_0, \boldsymbol{a}_1) + \boldsymbol{c}_0)$ set $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$		
	$\xrightarrow{\mathsf{ct}}$	
		$\alpha \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\alpha}$	
set $\boldsymbol{b}_1 = \alpha \cdot \boldsymbol{a}_0 - \boldsymbol{b}_0$ set $\boldsymbol{c}_1 = \alpha \cdot \mathcal{P}(\boldsymbol{a}_0) - \boldsymbol{c}_0$		
	$\xrightarrow{(\boldsymbol{b}_1, \boldsymbol{c}_1)}$	
		$\mathsf{ch} \xleftarrow{\$} \{0, 1\}$
	$\xleftarrow{\mathsf{ch}}$	

if $\mathsf{ch} = 0$, set $\mathsf{rs} = \boldsymbol{a}_0$
if $\mathsf{ch} = 1$, set $\mathsf{rs} = \boldsymbol{a}_1$		

	$\xrightarrow{\mathsf{rs}}$	

		if $\mathsf{ch} = 0$, check whether
		$\mathsf{ct}_0 \overset{?}{=} \mathsf{Commit}\left(\boldsymbol{a}_0 \| \alpha \cdot \boldsymbol{a}_0 - \boldsymbol{b}_1 \| \alpha \cdot \mathcal{P}(\boldsymbol{a}_0) - \boldsymbol{c}_1\right)$
		if $\mathsf{ch} = 1$, check whether
		$\mathsf{ct}_1 \overset{?}{=} \mathsf{Commit}\left(\boldsymbol{a}_1 \| \alpha \cdot (\boldsymbol{v} - \mathcal{P}(\boldsymbol{a}_1) + \mathcal{P}(\boldsymbol{0})) - \mathsf{G}(\boldsymbol{b}_1, \boldsymbol{a}_1) - \boldsymbol{c}_1\right)$

Figure 5: Illustration of 5-pass IDS of Sakumoto et al. [SSH11].

The authors showed in [SSH11] that their 5-pass IDS is statistical zero-knowledge against any cheating verifier. Hence, we have the following result.

Lemma B.4 (Statistical HVZK)**.** *The 5-pass IDS presented in Figure 5 is statistical HVZK if the underlying commitment scheme has statistical hiding property.*

Lemma B.5 (Extractor)**.** *Assume that the underlying commitment scheme is computationally binding. Then, the 5-pass IDS Figure 5 satisfies 3-special soundness (cf. Definition 2.14).*

[10]In fact, a formal reduction can be given such that if $\mathcal{A}$ cheats with probability $2/3+\epsilon$, then we can break either MQ-problem or the computational binding property of the underlying commitment scheme.

Proof. The proof essentially follows from that of Theorem 5 of [SSH11]. □

Lemma B.6 (Knowledge error)**.** *Suppose MQ-problem is intractable and the underlying commitment scheme is computationally binding. Then, the 5-pass IDS presented in [SSH11] is sound (cf. Definition 2.10) with knowledge error* $\frac{1}{2} + \frac{1}{2q}$.

Proof. For proof, see [CHR+16, Theorem 3.1]. □

C Omitted Proofs of Splitting Lemmas/Forking Abstractions

The following proposition is required for the proof of our special splitting lemma (see Lemma 3.8).

Proposition C.1. *Let X, Y, Z and U be finite sets. Let $B \subset X \times Y \times Z \times U$ be such that*

$$\Pr_{(x,y,z,u) \xleftarrow{\$} X\times Y\times Z\times U} [(x, y, z, u) \in B] = \varepsilon.$$

For any $\alpha \leq \varepsilon$, define

$$B^X_\alpha = \left\{ (x, y, z, u) \in X \times Y \times Z \times U : \Pr_{(y',z',u') \xleftarrow{\$} Y\times Z\times U} \big[(x, y', z', u') \in B\big] \geq \varepsilon - \alpha \right\}. \tag{32}$$

Then, the following statements hold for any $\alpha \leq \varepsilon$:

(i) $\Pr_{(x,y,z,u) \xleftarrow{\$} X\times Y\times Z\times U} \big[(x, y, z, u) \in B^X_\alpha\big] \geq \alpha$

(ii) $\forall (x, y, z, u) \in B^X_\alpha$*:* $\Pr_{(y',z',u') \xleftarrow{\$} Y\times Z\times U} [(x, y', z', u') \in B] \geq \varepsilon - \alpha$

(iii) $\Pr_{(x,y,z,u) \xleftarrow{\$} X\times Y\times Z\times U} \big[(x, y, z, u) \in B^X_\alpha \mid (x, y, z, u) \in B\big] \geq \alpha/\varepsilon.$

Proof. Follows immediately by setting Y as $Y \times Z \times U$ in the proof of Lemma 3.1. □

C.1 Proof of Special (1,2)-Multi-Splitting Lemma 3.8

Proof. Let 1-Evt and Evt denote the events "$(x, y, z, u) \in B \wedge (x, y, z', u') \in B$" and "$(x, y, z, u) \in B \wedge (x, y, z', u') \in B \wedge (x, y', z, u'') \in B$", respectively. Then, we get, Then, we get,

$$\begin{aligned}
\Delta_1 &= \Pr_{\substack{(x,y,z,u) \xleftarrow{\$} X\times Y\times Z\times U \\ (y',z',u',u'',u''') \xleftarrow{\$} Y\times Z\times U\times U\times U}} \big[(x, y', z', u''') \in B \mid \mathsf{Evt}\big] \\
&\geq \Pr_{\substack{(x,y,z,u) \xleftarrow{\$} X\times Y\times Z\times U \\ (y',z',u',u'',u''') \xleftarrow{\$} Y\times Z\times U\times U\times U}} \big[(x, y', z', u''') \in B \wedge (x, y, z, u) \in B^X_\alpha \mid \mathsf{Evt}\big] \\
&= \Delta_{11} \cdot \Delta_{12},
\end{aligned}$$

where B_α^X is as defined in Proposition C.1,

$$\begin{aligned}
\Delta_{11} &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}} \big[(x,y',z',u''') \in B \mid (x,y,z,u) \in B_\alpha^X \wedge \mathsf{Evt}\big] \\
&\geq \varepsilon - \alpha \qquad \text{[follows from item (ii) of Proposition C.1]}
\end{aligned}$$

and

$$\begin{aligned}
\Delta_{12} &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}} \big[(x,y,z,u) \in B_\alpha^X \mid \mathsf{Evt}\big] && (33)\\
&\geq \Pr_{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U} \big[(x,y,z,u) \in B_\alpha^X \mid (x,y,z,u) \in B\big] && (34)\\
&\geq \alpha/\varepsilon \qquad \text{[follows from item (iii) of Proposition C.1].}
\end{aligned}$$

Similarly to Claim 3.2, "$(x,y,z',u') \in B \wedge (x,y',z,u'') \in B$" increases the likelihood of "$(x,y,z,u) \in B_\alpha^X$". This fact is used in the transition from equation (33) to equation (34).

So, we can write

$$\Delta_1 \geq (\varepsilon - \alpha) \cdot \alpha/\varepsilon. \tag{35}$$

Now, we want to calculate the following conditional probability.

$$\begin{aligned}
\Delta_2 &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (y',z',u',u'')\xleftarrow{\$}Y\times Z\times U\times U}} \big[(x,y',z,u'') \in B \mid \text{1-}\mathsf{Evt}\big] \\
&\geq \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (y',z',u',u'')\xleftarrow{\$}Y\times Z\times U\times U}} \big[(x,y',z,u'') \in B \wedge (x,y,z',u') \in B_\alpha^X \mid \text{1-}\mathsf{Evt}\big] \\
&= \Delta_{21} \cdot \Delta_{22},
\end{aligned}$$

where B_α^X is as defined in Proposition C.1,

$$\begin{aligned}
\Delta_{21} &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (y',z',u',u'')\xleftarrow{\$}Y\times Z\times U\times U}} \big[(x,y',z,u'') \in B \mid (x,y,z',u') \in B_\alpha^X \wedge \text{1-}\mathsf{Evt}\big] \\
&\geq \varepsilon - \alpha \qquad \text{[follows from item (ii) of Proposition C.1]}
\end{aligned}$$

and

$$\begin{aligned}
\Delta_{22} &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (z',u')\xleftarrow{\$}Z\times U}} \big[(x,y,z',u') \in B_\alpha^X \mid \text{1-}\mathsf{Evt}\big] && (36)\\
&\geq \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U \\ (z',u')\xleftarrow{\$}Z\times U}} \big[(x,y,z',u') \in B_\alpha^X \mid (x,y,z',u') \in B\big] && (37)\\
&\geq \alpha/\varepsilon \qquad \text{[follows from item (iii) of Proposition C.1].}
\end{aligned}$$

Note that from equation (36) to equation (37), we use the fact that "$(x,y,z,u) \in B$" increases the likelihood of "$(x,y,z',u') \in B_\alpha^X$".

So, we can write

$$\Delta_2 \geq (\varepsilon - \alpha) \cdot \alpha/\varepsilon. \tag{38}$$

Finally, we have

$$\begin{aligned}
\Delta &= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}}\big[(x,y,z,u)\in B \wedge (x,y,z',u')\in B \wedge (x,y',z,u'')\in B \wedge (x,y',z',u''')\in B\big]\\
&= \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}}\big[(x,y',z,u'')\in B \mid \mathsf{Evt}\big] \cdot \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (y',z',u',u'',u''')\xleftarrow{\$}Y\times Z\times U\times U\times U}}[\mathsf{Evt}]\\
&= \Delta_1 \cdot \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (y',z',u',u'')\xleftarrow{\$}Y\times Z\times U\times U}}\big[(x,y',z,u'')\in B \mid \text{1-}\mathsf{Evt}\big] \cdot \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (z',u')\xleftarrow{\$}Z\times U}}[\text{1-}\mathsf{Evt}]\\
&= \Delta_1 \cdot \Delta_2 \cdot \Pr_{\substack{(x,y,z,u)\xleftarrow{\$}X\times Y\times Z\times U\\ (z',u')\xleftarrow{\$}Z\times U}}\big[(x,y,z,u)\in B \wedge (x,y,z',u')\in B\big]\\
&\geq (\varepsilon-\alpha)\cdot(\alpha/\varepsilon)\cdot(\varepsilon-\alpha)\cdot(\alpha/\varepsilon)\cdot(\varepsilon-\alpha)\cdot\alpha \qquad \text{[using equations (35), (38) and Lemma 3.3]}\\
&= (\varepsilon-\alpha)^3\cdot\alpha^3/\varepsilon^2.
\end{aligned}$$

□

C.2 Proof of (2,2)-Multi-Splitting Lemma 3.7

Proof. Let (1-2)-Evt denote the event "$(x,y,z)\in B \wedge (x,y',z')\in B \wedge (x,y,z''')\in B \wedge (x,y,z'''')\in B$". First, calculate the following conditional probability.

$$\begin{aligned}
\Delta_1 &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ (y'',z'')\xleftarrow{\$}Y\times Z\\ z'''\xleftarrow{\$}Z\\ z''''\xleftarrow{\$}Z}}\big[(x,y'',z'')\in B \mid \text{(1-2)-}\mathsf{Evt}\big]\\
&\geq \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ (y'',z'')\xleftarrow{\$}Y\times Z\\ z'''\xleftarrow{\$}Z\\ z''''\xleftarrow{\$}Z}}\big[(x,y'',z'')\in B \wedge (x,y,z)\in B_\alpha^X \mid \text{(1-2)-}\mathsf{Evt}\big]\\
&= \Delta_{11}\cdot\Delta_{12},
\end{aligned}$$

where B_α^X is as defined in Proposition 3.5,

$$\begin{aligned}
\Delta_{11} &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\ (y',z')\xleftarrow{\$}Y\times Z\\ (y'',z'')\xleftarrow{\$}Y\times Z\\ z'''\xleftarrow{\$}Z\\ z''''\xleftarrow{\$}Z}}\big[(x,y'',z'')\in B \mid (x,y,z)\in B_\alpha^X \wedge \text{(1-2)-}\mathsf{Evt}\big]\\
&\geq \varepsilon-\alpha \qquad \text{[follows from item (ii) of Proposition 3.5]}
\end{aligned}$$

and

$$\begin{aligned}
\Delta_{12} &= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\(y'',z'')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}\left[(x,y,z)\in B_\alpha^X \mid \textsf{(1-2)-Evt}\right]\\
&= \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}\left[(x,y,z)\in B_\alpha^X \mid {\scriptstyle (x,y,z)\in B\wedge(x,y',z')\in B\wedge \atop (x,y,z''')\in B\wedge(x,y,z'''')\in B}\right] \qquad (39)\\
&\geq \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}\left[(x,y,z)\in B_\alpha^X \mid (x,y,z)\in B\right] \qquad (40)\\
&\geq \alpha/\varepsilon \qquad \text{[follows from item (iii) of Proposition 3.5].}
\end{aligned}$$

Note that from equation (39) to equation (40), we use the fact that "$(x,y',z')\in B\wedge(x,y,z''')\in B\wedge(x,y,z'''')\in B$" increases the likelihood of "$(x,y,z)\in B_\alpha^X$".

So, we can write

$$\Delta_1 \geq (\varepsilon-\alpha)\cdot\alpha/\varepsilon. \qquad (41)$$

Finally, we have

$$\begin{aligned}
&\Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\(y'',z'')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}\left[(x,y,z)\in B\wedge(x,y',z')\in B\wedge(x,y'',z'')\in B\wedge(x,y,z''')\in B\wedge(x,y,z'''')\in B\right]\\
=\ &\Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\(y'',z'')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}\left[(x,y'',z'')\in B \mid \textsf{(1-2)-Evt}\right]\cdot \Pr_{\substack{(x,y,z)\xleftarrow{\$}X\times Y\times Z\\(y',z')\xleftarrow{\$}Y\times Z\\z'''\xleftarrow{\$}Z\\z''''\xleftarrow{\$}Z}}[\textsf{(1-2)-Evt}]\\
\geq\ &(\varepsilon-\alpha)\cdot(\alpha/\varepsilon)\cdot(\varepsilon-\alpha)^3\cdot\alpha^3/\varepsilon^2 \qquad \text{[using equation (41) and Lemma 3.6]}\\
=\ &(\varepsilon-\alpha)^4\cdot\alpha^4/\varepsilon^3.
\end{aligned}$$

□

C.3 Proof of Corollary 3.12

Proof. First, note that

$$\textsf{acc} = \sum_{\substack{j_1,j_2\in[\nu]\\j_1<j_2}} \textsf{acc}(\mathcal{W}_{j_1,j_2}).$$

Then, we calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk} &= \sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{frk}(\mathcal{W}_{j_1,j_2}, j_1, j_2) \geq \sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{acc}(\mathcal{W}_{j_1,j_2}) \cdot \left(\frac{\mathsf{acc}^4(\mathcal{W}_{j_1,j_2})}{256} - \frac{6}{|\mathrm{R}|}\right) \qquad \text{[using Lemma 3.11]}\\
&= \left(\sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \frac{\mathsf{acc}^5(\mathcal{W}_{j_1,j_2})}{256}\right) - \frac{6\cdot\mathsf{acc}}{|\mathrm{R}|} \geq \frac{16}{256\cdot(\nu^2-\nu)^4}\left(\sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{acc}(\mathcal{W}_{j_1,j_2})\right)^5 - \frac{6\cdot\mathsf{acc}}{|\mathrm{R}|}\\
&= \frac{\mathsf{acc}^5}{16\cdot(\nu^2-\nu)^4} - \frac{6\cdot\mathsf{acc}}{|\mathrm{R}|} = \mathsf{acc}\cdot\left(\frac{\mathsf{acc}^4}{16\cdot(\nu^2-\nu)^4} - \frac{6}{|\mathrm{R}|}\right),
\end{aligned}$$

where we use Proposition A.1 with $a = 5$ in the above intermediate inequality. This completes the proof. □

C.4 Proof of Special (1,2)-Multi-Splitting-Forking Lemma 3.13

Proof. Let SEvt denote the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}'') \in \mathcal{C} \wedge (I, \omega, \boldsymbol{h}''') \in \mathcal{C}$". Set $X = \mathcal{I} \times \Omega \times \mathrm{R}^{j_1-1}$, $Y = \mathrm{R}^{j_2-j_1}$, $Z = \mathrm{R}$ and $U = \mathrm{R}^{\nu-j_2}$. So, we can write the sample space as $\mathcal{I} \times \Omega \times \mathsf{R}^\nu = X \times Y \times Z \times U$. Applying Lemma 3.8 with $\varepsilon = \mathsf{acc}(\mathcal{C})$ and $\alpha = \varepsilon/2$, we have

$$\Pr_{\substack{(I,\omega,\boldsymbol{h})\xleftarrow{\$}\mathcal{I}\times\Omega\times\mathrm{R}^\nu\\ \boldsymbol{h}'\longleftarrow\mathrm{R}^\nu|\boldsymbol{h}_{[j_2-1]}\\ \boldsymbol{h}''\longleftarrow\mathrm{R}^\nu|\boldsymbol{h}_{[j_1-1]}\ \text{s.t}\ h''_{j_2}=h_{j_2}\\ \boldsymbol{h}'''\longleftarrow\mathrm{R}^\nu|\boldsymbol{h}''_{[j_2-1]}\ \text{s.t}\ h'''_{j_2}=h'_{j_2}}}[\mathsf{SEvt}] \geq \frac{\mathsf{acc}^4(\mathcal{C})}{64}.$$

Let HEvt denote the event "$h'_{j_2} \neq h_{j_2} \wedge h''_{j_1} \neq h_{j_1}$". Then, we can calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk}(\mathcal{C}, j_1, j_2) &= \Pr[\mathsf{SEvt} \wedge \mathsf{HEvt}]\\
&= \Pr[\mathsf{SEvt}] - \Pr\left[\mathsf{SEvt} \wedge \overline{\mathsf{HEvt}}\right]\\
&\geq \Pr[\mathsf{SEvt}] - \Pr\left[(I, \omega, \boldsymbol{h}) \in \mathcal{C} \wedge \overline{\mathsf{HEvt}}\right] \qquad (42)\\
&\geq \mathsf{acc}^4(\mathcal{C})/64 - \Pr[(I, \omega, \boldsymbol{h}) \in \mathcal{C}] \cdot \Pr\left[\overline{\mathsf{HEvt}}\right] \qquad (43)\\
&\geq \mathsf{acc}^4(\mathcal{C})/64 - \mathsf{acc}(\mathcal{C}) \cdot \Pr\left[h'_{j_2} = h_{j_2} \vee h''_{j_1} = h_{j_1}\right]\\
&\geq \mathsf{acc}^4(\mathcal{C})/64 - \mathsf{acc}(\mathcal{C}) \cdot 2/|\mathrm{R}| \qquad \text{(using union bound)}\\
&= \mathsf{acc}(\mathcal{C}) \cdot \left(\frac{\mathsf{acc}^3(\mathcal{C})}{64} - \frac{2}{|\mathrm{R}|}\right).
\end{aligned}$$

Note that from equation (42) to equation (43), we use the fact that the event "$(I, \omega, \boldsymbol{h}) \in \mathcal{C}$" and $\overline{\mathsf{HEvt}}$ are independent as $h_{j_1}, h_{j_2}, h''_{j_1}$ and h'_{j_2} are chosen independently. This completes the proof. □

C.5 Proof of Corollary 3.14

Proof. First, note that

$$\mathsf{acc} = \sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{acc}(\mathcal{W}_{j_1,j_2}).$$

Then, we calculate the desired probability as follows.

$$\begin{aligned}
\mathsf{frk} &= \sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{frk}(\mathcal{W}_{j_1,j_2}, j_1, j_2) \geq \sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{acc}(\mathcal{W}_{j_1,j_2}) \cdot \left(\frac{\mathsf{acc}^3(\mathcal{W}_{j_1,j_2})}{64} - \frac{2}{|\mathrm{R}|}\right) && \text{[using Lemma 3.13]}\\
&= \left(\sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \frac{\mathsf{acc}^4(\mathcal{W}_{j_1,j_2})}{64}\right) - \frac{2\cdot\mathsf{acc}}{|\mathrm{R}|} \geq \frac{8}{64\cdot(\nu^2-\nu)^3}\left(\sum_{\substack{j_1,j_2\in[\nu]\\ j_1<j_2}} \mathsf{acc}(\mathcal{W}_{j_1,j_2})\right)^4 - \frac{2\cdot\mathsf{acc}}{|\mathrm{R}|}\\
&= \frac{\mathsf{acc}^4}{8\cdot(\nu^2-\nu)^3} - \frac{2\cdot\mathsf{acc}}{|\mathrm{R}|} = \mathsf{acc}\cdot\left(\frac{\mathsf{acc}^3}{8\cdot(\nu^2-\nu)^3} - \frac{2}{|\mathrm{R}|}\right),
\end{aligned}$$

where we use Proposition A.1 with $a = 4$ in the above intermediate inequality. This completes the proof. □